UNCLASSIFIED

AD NUMBER
ADB075938
LIMITATION CHANGES
TO: **Approved for public release; distribution is unlimited.**
FROM: **Distribution authorized to U.S. Gov't. agencies only; Test and Evaluation; JUN 1983. Other requests shall be referred to Defense Advanced Research Projects Agency, TIO, 1400 Wilson Blvd, Arlington, VA 22209.**
AUTHORITY
darpa ltr, 21 sep 1983

THIS PAGE IS UNCLASSIFIED

AD B075938

AUTHORITY: DARPA

ltr 21 Sep 83

AD B075938

DARPA

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

Improved Packet Radio (IPR) Operating System User Guide

Rockwell International

DTIC
ELECTE
AUG 24 1983
S E D

1400 WILSON BOULEVARD ARLINGTON, VIRGINIA 22209

DTIC FILE COPY

83 08 16 141

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-B0[illegible]
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) IMPROVED PACKET RADIO (IPR) OPERATING SYSTEM USER GUIDE	5. TYPE OF REPORT & PERIOD COVERED PACKET RADIO SOFTWARE OPERATING SYSTEM USER GUIDE
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) DICK SUNLIN, JOHN JUBIN, TONY CARUSO, TIM QUILICI, NEIL GOWER, HANK HARWELL	8. CONTRACT OR GRANT NUMBER(s) MDA903-79-C-0088
9. PERFORMING ORGANIZATION NAME AND ADDRESS ROCKWELL INTERNATIONAL COLLINS COMMUNICATIONS SYSTEMS DIVISION 3200 E. RENNER ROAD, RICHARDSON, TX 75081	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS AO 3682
11. CONTROLLING OFFICE NAME AND ADDRESS DEFENSE ADVANCED RESEARCH PROJECT AGENCY 1400 WILSON BLVD. ARLINGTON, VA 22209	12. REPORT DATE 30 JUNE 1983
	13. NUMBER OF PAGES 114
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION LIMITED TO U.S. GOVERNMENT Agencies ~~ORGANIZATIONS~~ ONLY T&E, OTHER REQUESTS FOR THIS DOCUMENT MUST BE REFERRED TO DARPA/TIO, 1400 WILSON BLVD., ARLINGTON, VA 22209.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

PACKET RADIO, (IPR), OPERATING SYSTEM USER GUIDE, SOFTWARE OPERATING SYSTEM, INTERFACE SPECIFICATIONS.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

THIS DOCUMENT DESCRIBES THE OPERATING SYSTEM SOFTWARE OF THE MULTIPROCESSOR IMPROVED PACKET RADIO UNIT (IPR). IT CONTAINS THE INTERFACE SPECIFICATION FOR DEVELOPERS OF IPR SYSTEM SOFTWARE.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

IMPROVED PACKET RADIO (IPR) OPERATING SYSTEM USERS GUIDE

by

Dick Sunlin
John Jubin
Tony Caruso
Tim Quilici
Neil Gower
Hank Harwell

Collins Communications Systems Division
Rockwell International

ABSTRACT

This document describes the operating system software utilized in the multiprocessor packet radio unit (IPR). It is intended as a users guide for IPR operators and as an operating system interface specification for developers of IPR applications processes.

This document was sponsored by the Defense Advanced Research Projects Agency under ARPA Order No. 3682, Contract No. MDA903-79-C-0088, and monitored by Dr. B. Leiner.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

DTIC COPY INSPECTED

Accession For
NTIS GRA&I
DTIC TAB X
Unannounced
Justification
By
Distribution/
Availability Codes
Avail and/or
Dist Special
B

CONTENTS

ILLUSTRATIONS

1 INTRODUCTION

This document describes the operating system software of the multiprocessor Improved Packet Radio unit (IPR). It is intended as a users guide for operation of the IPR and as an operating system interface specification for developers of IPR system software.

This document consists of four major sections. Section 2 provides an overview description of the IPR digital unit hardware. Section 3 describes the operating system software. Section 4 defines the structure and organization of the IPR user processes (jobs) and user process interfaces to the operating system software. Section 5 describes IPR operating procedures. This document, together with the separate specifications of the IPR user processes, provides a comprehensive description of the IPR system software.

Throughout this document, the most significant (left-most) bit of each word is numbered 0 while the least significant (right-most) bit is numbered 15. This is the numbering convention adopted by Texas Instruments.

2 HARDWARE ARCHITECTURE

The hardware configuration of the IPR is illustrated in Figure 1. It consists of two microprocessors, memory (both PROM and RAM), DMA I/O interfaces to the Radio and 1822 interfaces, a hexadecimal display and an RS-232 interface to a console. All devices are connected to a common bus consisting of 20 address lines, 16 data lines, and appropriate control signals.

The bus coupler may be implemented to expand the system to a dual bus structure. The device connected to the bus coupler may contain additional CPUs, memory, and peripheral devices.

2.1 Microprocessor

Each of the CPUs in the IPR is a Texas Instruments SBP 9900 microprocessor using I^2L technology. The 9900 is a 16-bit CPU that implements 69 different instructions (including fixed point multiply and divide) using seven addressing modes. When the 9900 is operated with a 3 MHz clock, instruction execution times (excluding multiply and divide) vary from 2.7 us to 20.0 us with an average of about 7 us. Instructions are one, two, or three 16 bit words in length.

The 9900 provides 16 vectored prioritized hardware interrupts. Instead of a stack architecture, the CPU performs "context switching" of the processor work space, program counter, and status registers. A "work space" is the CPU's 16 general purpose registers which are resident in the system's main memory (rather than on the CPU chip itself). When a system call, subroutine call, or interrupt occurs, a context switch is executed by saving a pointer to the old work space, program counter, and status in a new work space used by the called process.

The 9900 contains a special serial I/O interface called the CRU (Communications Register Unit) that is utilized in the IPR to interface with the limit and bias registers.

```
-----------   -----------   -----------   -----------   -----------
|   CPU   |   |   CPU   |   | ROM/RAM |   | DISPLAY |   | CONSOLE |
-----------   -----------   | MEMORY  |   |  PANEL  |   -----------
  |    |        |    |      -----------   -----------        |
  |    |        |    |        |    |           |        -----------
  |  -----      |  -----      |  -----    -----------   | CONSOLE |
  |  |ADR|      |  |ADR|      |  |ADR|    | CONFIG/ |   |   I/F   |
  |  |MAP|      |  |MAP|      |  |DEC|    | DISPLAY |   |INTERVAL |
  |  -----      |  -----      |  -----    |   I/F   |   | TMR/CLK |
  |    |        |    |        |    |      -----------   -----------
  |    |        |    |        |    |           |             |
-----------   -----------   -----------   -----------   -----------
| MASTER  |   | MASTER  |   |  SLAVE  |   |  SLAVE  |   |  SLAVE  |
-----------   -----------   -----------   -----------   -----------
     |             |             |             |             |
-------------------------------------------------------------------
                              BUS
-------------------------------------------------------------------
                               |
                          ----------
                          | MASTER |
                          ----------
                               |
          ----------------------------------------------
          |          |         |          |          |
      --------   --------   --------   --------   --------
      | 1822 |   | 1822 |   |RADIO |   |RADIO |   |RADIO |
      |  RX  |   |  TX  |   | RX1  |   | RX2  |   |  TX  |
      | DMA  |   | DMA  |   | DMA  |   | DMA  |   | DMA  |
      | I/F  |   | I/F  |   | I/F  |   | I/F  |   | I/F  |
      --------   --------   --------   --------   --------
          |          |         |          |          |
      -------------------   ---------------------------
      |    STATION      |   |          RADIO          |
      -------------------   ---------------------------
```

FIGURE 1 FUNCTIONAL BLOCK DIAGRAM OF THE IPR HARDWARE

2.2 Address Mapping

The IPR's digital unit implements a 20 bit address bus to provide 2^{20}, or 1024K, unique word addresses. The 9900 CPU is able to address 2^{16} unique bytes of memory or peripherals. However, the CPU only outputs a 15 bit address bus and all memory/peripheral operations are performed on words. (The least significant address bit is used internally by the CPU for byte addressing.) All 15-bit addresses output by the CPU are modified by the IPR mapping hardware to provide a 20-bit bus address. This is accomplished by summing the CPU address with one of several bias registers as shown in Figure 2. The particular bias register used in the summing process is determined by the contents of the limit registers.

Associated with each CPU in the IPR are three software variable bias registers, one fixed bias (hardware strapped) register and three software variable limit registers. These registers are set and read under software control via the CPU's CRU interface. The three limit registers, each 7 bits in length, may be used to partition the 32K address space of the CPU into four zones. The 7-bit limit registers are compared with the most significant 7 bits of address output by the CPU. If the CPU address is equal to, or greater than, limit register 3, then zone 4 is defined and the CPU address is summed with fixed bias 4. If the CPU address is not in zone 4 and is equal to, or greater than, limit register 2, then zone 3 is defined and the CPU address is summed with variable bias 3. If the CPU address is not in zone 4 or zone 3 and is equal to, or greater than, limit register 1, then zone 2 is defined and the CPU address is summed with variable bias 2. If the CPU address is not in zones 2, 3, or 4, then zone 1 is defined and the CPU address is summed with variable bias 1. The length of each zone is determined by the content of the limit registers.

When power is first applied to the CPU, and before the CPU has had time to define the limit and bias registers, a special mechanism must be employed to enable a default form of mapping to occur. In the normal mapping mode, the CPU address is simultaneously compared to all three limit registers and the zone decision (based upon the rules stated above) is presented to the bias registers in the form of two signal lines. The four possible states of these two lines determine which of the four bias registers will be added to the CPU address. However, when power is first applied to the system, a flip-flop is set on the mapping board which bypasses the limit registers and forces the default mapping to occur. When the flip-flop is set, the two signal lines to the bias registers are set high forcing the zone 4 bias register to be used for all CPU addresses. While the bias registers for zones 1, 2, and 3 are software controllable, the zone 4 bias register is strapped in hardware to be 1111100000_2. In this way the default flip-flop and hardware strapped bias 4 allows the system to begin executing instructions at a known location in the 1024K system address space. The default flip-flop, and thus the default mapping, stays set until the CPU writes to limit register 3.

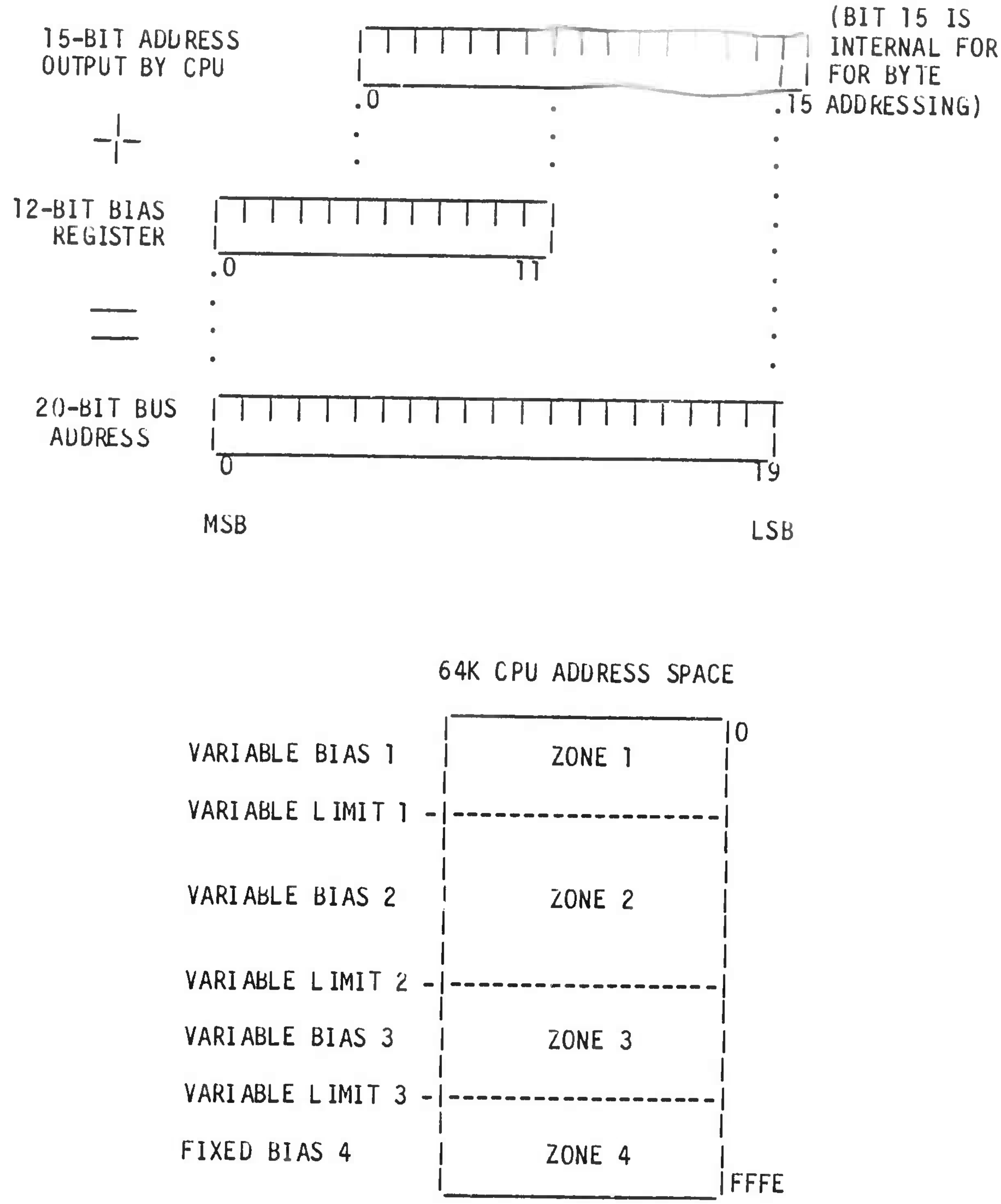


FIGURE 2 ADDRESS MAPPING WITH CPU BIAS AND LIMIT REGISTERS

The comparison of a CPU address with a limit value is effectively done by subtracting the limit value from the address and testing for a borrow out of the most significant bit. Because the mapping board was designed to perform additions, rather than subtractions, the limit register must be loaded with the two's complement of the intended limit value. The comparison of the CPU address with the limit value is then performed by adding the CPU address to the limit register and testing for a carry out.

2.3 Memory

The IPR utilizes read only (PROM) and random access (RAM) memory. Parity is generated and checked on each 16 bit PROM or RAM word. Most of the IPR operating system resides in PROM. The remainder of the operating system, and all of the user processes, are resident in RAM. Figure 3 lists the address at which memory exists in the IPR.

20 BIT BUS ADDRESS (BASE 16)	16 BIT CPU ADDRESS (BASE 16) (before biasing)	IMPLEMENTATION
00000 - 02BFF	0000 - 55FE	11K WORDS RAM
04000 - 04FFF	8000 - 9FFE	4K WORDS RAM
FD400 - FDFFF	A800 - BFFE	3K WORDS RAM
FEC00 - FEFFF	D800 - DFFE	1K PERIPHERAL
FF000 - FFFFF	E000 - FFFE	4K WORDS PROM

FIGURE 3 IPR MEMORY ORGANIZATION

2.4 Interrupt Assignments

The 9900 CPU implements 16 interrupt levels with level 0 being the highest and level 15 being the lowest. When the level of a pending interrupt is less than, or equal to, the enabling mask level (higher or equal priority interrupt), the CPU recognizes the interrupt and initiates a context switch following completion of the currently executing instruction. The processor fetches the new context workspace pointer (WP) and program counter (PC) from the appropriate interrupt vector in the CPU's zone 1. Then, the previous context WP, PC, and status are stored in workspace registers R13, R14, and R15, respectively, of the new workspace. The CPU then forces the interrupt mask to a value that is one less than the level of the interrupt being serviced. (When a level 0 interrupt is being serviced, the mask is set to 0.) This allows only interrupts of higher priority to interrupt a service routine. The IPR's interrupt vector assignments are shown in Figure 4.

The specific interrupts are unique to each CPU. Each CPU may independently be interrupted by the specific interrupt source. Common interrupts generate an interrupt to both CPUs from a single interrupt source. The operating system software resolves contention so that only one CPU performs the interrupt servicing of the common interrupts.

A failure interrupt is generated whenever a bus response timeout from memory is detected, an invalid instruction operation is detected, a memory parity error occurs, or the "watch dog timer" interval has elapsed.

A CPU directed interrupt is generated by operating system software. A CPU can interrupt itself or any other CPU in the system.

The interval timer decrements the 16 bit value loaded by the software and generates an interrupt when the value reaches zero. Each count of the interval timer equals 1.667 msec.

An interrupt is generated by the Radio or 1822 transmit or receive DMA's upon completion of a DMA data transfer.

2.5 Extended Operations (XOPs)

The extended operation (XOP) instruction is used for interfacing with the operating system software. Several vectors, similar to the interrupt vectors, are defined for specific operating system functions. The XOP vector assignments are shown in Figure 5.

2.6 Special Purpose Instructions

The CKON, CKOF, RSET and ABS instructions are utilized by the IPR to perform special system functions. While the ABS instruction may be used by any user job to resolve contention, the CKON, CKOF and RSET instructions are reserved for use by the operating system.

2.6.1 CKON/CKOF

The CKON instruction is used to enable and restart the CPU "watch dog timer". The watch dog timer interval is approximately 2 seconds. The CKOF instruction disables the watch dog timer. Each IPR CPU implements a watch dog timer.

VECTOR	CPU	FUNCTION
0	Specific	Reset (Unused)
1	Specific	Failure; Bus Response Timeout, Invalid Op-Code, Memory Parity Error, Watch Dog Timer
2		Unused
3	Specific	CPU Directed
4	Common	Interval Timer
5		Unused
6	Common	Console I/O
7	Common	Radio Receive Channel 1
8	Common	Radio Receive Channel 2
9	Common	Radio Transmit
10		Unused
11	Common	1822 Receive
12	Common	1822 Transmit
13		Unused
14		Unused
15		Unused

FIGURE 4 IPR INTERRUPT VECTOR ASSIGNMENTS

VECTOR	USE
0	Job Control (Scheduler)
1	Job Call
2	Trace Breakpoint
3	Buffer Management
4	Console I/O
5	Radio/1822 DMA I/O
6	Unused
7	Utility
8	1822 Ready
9	Down Load
10	I/O Tags
11-15	Unused

FIGURE 5 XOP VECTOR ASSIGNMENTS

2.6.2 RSET

The RSET instruction invokes restart initialization of all CPUs in the IPR.

2.6.3 ABS

The ABS instruction is used to resolve contention for commonly used processes or data structures. The ABS instruction generates the absolute value of a specified signed two's complement operand. The

CPU's status word is modified based on the value of the operand before the absolute value is calculated and stored. Also, special hardware in the IPR recognizes when the ABS instruction is being executed and makes the operand fetch/modify/store operations indivisible. That is, once access to the bus has been granted to fetch the operand, the bus is not released until the absolute value of the operand has been stored. Thus, all other CPUs are inhibited from gaining access to the operand during the execution of the ABS instruction. These features of the ABS instruction allow the creation of flags to resolve contention between jobs for a common data structure. Such a contention flag is nothing more than a location in memory that contains alternately positive and negative numbers. (In the IPR, plus and minus one are typically used.) A negative value in a contention flag indicates the associated data structure is not being used by another job. To gain access to the data, a job performs an ABS instruction on the contention flag and the flag is left with a positive value. If the status word indicates the flag was negative, then the job knows the data was not being used and is now free to access the data. If, after performing the ABS instruction on the flag, the status word indicates the flag was already positive, then the job knows the data is already being accessed by another job.

2.7 Direct Memory Access (DMA) Interfaces

2.7.1 Radio DMA I/O

The radio transmit/receive channels implement the IPR digital unit interface with the IPR radio. The operating system software enables the radio reception of a packet by enabling one or both of the radio receive DMA channels. Either radio receive DMA channel (if enabled) may receive a 100 or 400 Kbit data packet. Thus, receiver blocking, the situation where a packet cannot be received for want of an enabled receive channel, is minimized in the IPR.

A radio packet transmit is initiated by the enabling of the radio transmit DMA channel. The transmit may occur immediately, in which case a packet currently being received will be lost, or the transmit may be delayed until the RF channel is sensed to be unused.

A CPU interrupt is generated upon completion of a radio packet transmission or reception.

2.7.2 1822 DMA I/O

The 1822 transmit/receive channels provide for full duplex transfer of packets to an attached host computer. The 1822 DMA channels are normally used to interconnect network terminals and stations to the IPR. A CPU interrupt is generated upon completion of an 1822 DMA packet transmission or reception.

2.8 Operator Console

The IPR will support an operator console connected to the I/O channel (RS-232) interface. A single console is time shared for communication with all software processes resident in the IPR.

2.9 Elapsed Time Clock and Interval Timer

The IPR implements a single 32 bit elapsed time clock. The least and most significant words of the clock may be read by IPR software processes for the determination of elapsed time. The least significant clock word has a tick rate of 6.51 usec, and the most significant clock word has a tick rate of 427 msec.

The IPR also implements a 16 bit interval timer. The timer may be loaded with any 16 bit binary count. The count is decremented at a rate of 1.667 msec and a CPU interrupt is generated when the count reaches zero. The interval timer is reserved for use by the operating system software.

2.10 Manual Control and Display

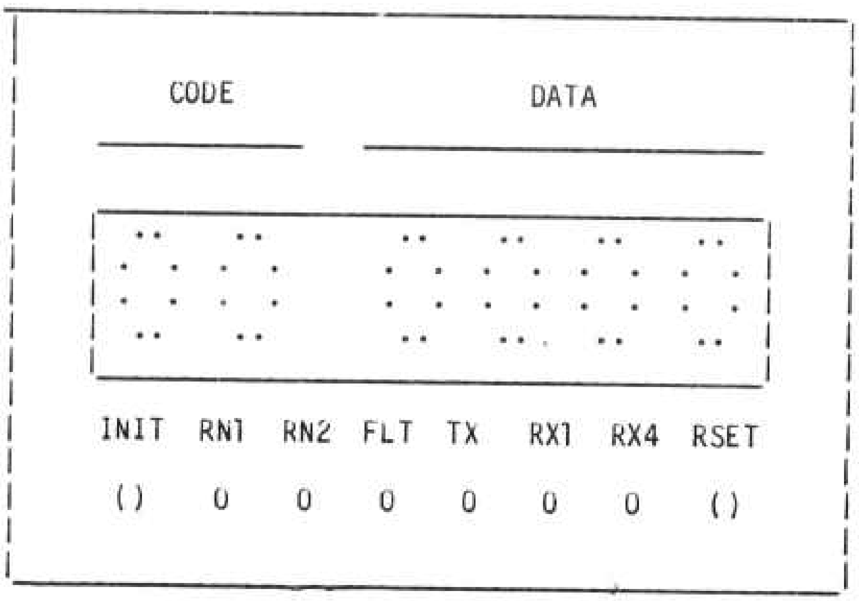
The control/display panel is illustrated in Figure 6. This panel is located on the front of the IPR digital unit under a hinged door.

The code and data hexadecimal (0-9, A-F) display may be set, blanked (turned off), or blinked by the PR software. It is utilized to display status and define failure during initialization and normal operation of the IPR.

The initialization (INIT) pushbutton switch is depressed to invoke power-up initialization of the PR. The restart (RSET) pushbutton switch is used to invoke restart initialization of the PR. The RSET pushbutton operation is functionally identical to the RSET instruction.

The remaining LED displays are directly driven by the hardware and are not under direct control of the PR software. The run indicators, when illuminated, indicate that the defined CPU is fetching instructions, and that it is not idled (IDLE instruction) or failed.

The fault indicator (FLT) is illuminated for approximately 2 seconds whenever a bus response timeout, invalid op-code, memory parity error, or elapsed watch dog timer error occurs.



LEGEND

CODE	Two-digit Hexadecimal Display
DATA	Four-digit Hexadecimal Display
INIT	Power-up Initialization Switch
RSET	Restart Initialization Switch
RN1	CPU 1 Run Indicator LED
RN2	CPU 2 Run Indicator LED
FLT	IPR Fault Indicator LED
TX	Radio Transmit Indicator LED
RX1	Radio 100kbps Receive Indicator LED
RX4	Radio 400kbps Receive Indicator LED

FIGURE 6 IPR MANUAL CONTROL/DISPLAY PANEL

The radio transmit indicator (TX) is lit for the duration of a packet transmission. The radio receive 100 Kbit (RX1), or 400 Kbit (RX4) receive indicators are lit for the duration of the packet reception.

2.11 Hardware Strapping Options

Hardware straps utilized by the IPR system software are located on the "Configuration/Display interface" and "I/O Channel and Interval Timer" printed circuit boards.

2.11.1 Configuration/Display Interface

Four DIP socket packages are located on the top left corner of this board when viewed with the backplane connector facing to the right. Each package contains eight straps. When a strap is inserted in the socket, the system software reads a binary zero.

2.11.1.1 PR ID

The two left-most strap packages define the PR's ID. The most significant bit of the 16-bit PR ID is the left-most strap of the two ID packages.

2.11.1.2 Auto-restart Enable

The left-most strap of the third strap package enables (1) or disables (0) auto-restart in the event of failure of the IPR.

2.11.1.3 Down-Line Load Enable

The second strap of the third strap package enables (1) or inhibits (0) the request for a down-line load by the IPR software during the IPR initialization.

2.11.1.4 Default RF Frequency

The right-most six straps of the third strap package define the default radio RF frequency utilized for down-line loading of the IPR. The binary value in the straps and the corresponding RF frequency are shown below.

STRAPS	RF FREQ (MHZ)
111001	1712.3
111000	1718.7
110111	1725.1
110110	1731.5
110101	1737.9
110100	1744.3
110011	1750.7
110010	1757.1
110001	1763.5
110000	1769.9
101001	1776.3
101000	1782.7
100111	1789.1
100110	1795.5
100101	1801.9
100100	1808.3
100011	1814.7
100010	1821.1
100001	1827.5
100000	1833.9
011001	1840.3

The six frequency straps are used to control a 6-bit BCD divide chain. The allowable strap values shown above represent BCD numbers ranging from 19_{10} to 39_{10}. If a BCD number less than 19 is used, the frequency generated will be too high and will be filtered out in the RF head. There are no BCD numbers higher than the legal 39. If the straps are set to a value that is not a valid BCD number, the frequency generated is indeterminate.

2.11.1.5 Initialization Control

The left-most five straps of the right-most strap package are available for applications software use. The remaining three straps are used for special maintenance and test options during initialization. All straps in this package are normally set to binary zero.

Strap 6	Loop on Error
Strap 7	Write Memory in Loop on Error
Strap 8	Do mini-initialization

2.11.2 I/O Channel and Interval Timer

Hardware strapping options are implemented on the I/O channel and Interval Timer PC board to define the type and BAUD rate of the terminal device connected to the RS-232 I/O channel port.

2.11.2.1 Terminal Type

One type of terminal that is commonly used with the IPR is the TI Silent 700 terminal. The Silent 700 is a hard-copy terminal that will transmit and receive characters at baud rates up to 1200, but mechanical reasons prevent the terminal from printing more than 30 characters per second. In order to accommodate use of the Silent 700 with the IPR, a slide switch is provided on the front edge of the I/O channel board. When this switch is in the down position, the hardware inserts a 25 msec delay between the Uart's "TX Ready" signal and the generation of an interrupt. This delay is sufficient to limit the effective character rate to 30 characters per second. When the slide switch is in the up position, no artificial delays are inserted by the hardware.

2.11.2.2 Terminal Baud Rate

The baud rate is specified in a rotary switch facing the front of the PC board. The switch positions and their corresponding baud rates are shown below.

SWITCH POSITION	BAUD RATE
0	110
1	150
2	300
3	2400
4	1200
5	1800
6	4800
7	9600
8	2400
9	600

While the IPR's UART is able to transmit and receive individual characters at all of these baud rates, the CPU is not fast enough to input a continuous stream of characters above 4800 baud. In other words, a user can communicate with the IPR at 9600 baud if the input is from a keyboard, but software can not be loaded at rates above 4800 baud.

3 OPERATING SYSTEM SOFTWARE DESCRIPTION

3.1 Overview

The two major goals of the operating system design were to optimize processing speed in the multiprocessor PR and provide a structure which permitted modular expansion of the IPR hardware and software. Maximum processing speed is obtained by maximizing the concurrent parallel execution of the PR processing tasks. The operating system's design provides for expansion to a multibus (multicluster) architecture with additional CPUs, memory, and peripheral devices.

The operating system implements a multiprogrammed event-driven processing environment for the PR processing tasks. Multiple independent user processes, hereafter called jobs, may be executed concurrently in the PR. Each CPU independently executes the operating system processes and system jobs. Each CPU contends with the other CPUs to process the system tasks. Processing tasks are not dedicated to CPUs and all CPUs have equal access to the PR system memory. Each CPU may periodically execute a system job and perform interrupt processing of the common interrupts. Separate common interrupts are processed simultaneously by the two CPUs.

Processing is scheduled in the PR based upon the first in/first out occurrence of an event and the relative execution priorities of these events. Four execution priorities are currently defined by the operating system. These are listed below in the order of highest to lowest priority:

Executive
Job Priority 1
Job Priority 2
Job Priority 3 (Debug only)
Idle

Each CPU's execution priority varies with time and equals the priority of the task it is currently executing. If a CPU finds no processing tasks scheduled, it will go idle (cease executing) until the occurrence of an interrupt. Idling the CPU is done to reduce system bus contention. When a processing task is scheduled, all CPUs are alerted via generation of an interrupt.

Jobs are scheduled for execution based upon the occurrence of an event. An event may be the completion of an I/O operation, passage of a time interval specified by a job, an inter-job call or an operator console command. An event is processed by executing a job. When the processing has completed, the job suspends execution until occurrence of another event. The operating system passes the job call event parameters between independent processes and saves/restores the machine state for each system job.

Jobs are created in the PR by loading of job object code remotely via the radio or 1822 interfaces or via the local console interface. Each job is preassigned a unique name and identification number. This is done to facilitate operator monitoring and control and interjob communications.

Operation of the PR may be unattended or totally under operator control via local console commands. These local console commands enable an operator to control and monitor PR operation plus provide capabilities for software debugging.

The operating system implements built in test (BIT) diagnostics executed when the PR is initialized or has failed. Auto-restart in the event of failure is provided.

The major processes of the PR operating system are illustrated in Figure 7 and briefly described below. They are defined in greater detail in the following paragraphs of Section 3.

POWER-UP INITIALIZATION/BIT DIAGNOSTICS - Initializes the PR from a never executed state. BIT diagnostics verify PROM contents, test mapping registers, and RAM memory. The remaining portion of the operating system is loaded and the operating system RAM is initialized.

RESTART INITIALIZATION/BIT DIAGNOSTICS - Reinitializes a PR which has failed or is restarting. All jobs are checksummed and reinitialized equivalent to the initial load of the jobs. Down-line loading is requested if auto-restart and down-line load hardware straps are enabled and no jobs are resident or a job code space checksum error is detected.

JOB CONTROL (MONITOR/SCHEDULE) - Includes all processes utilized to monitor and control job execution. These include the: scheduler, interjob call, interval timer, clock queue service, CPU directed interrupt, PR failure interrupt (bus timeout, memory parity error, invalid instruction code, watch dog timer).

CONTROL/MONITOR (DEBUG) JOB - Operating system process executed as a system job. It implements operator console commands utilized to monitor and control job execution and provide software debug aids.

PROCESSES

- Power-Up Initialization/BIT Diagnostics
- Restart Initialization/BIT Diagnostics
- Job Control (Scheduler/Monitor/Fault Intercept)
- Radio/1822 DMA I/O
- Console (I/O Channel) I/O
- Buffer Management
- Remote/Local Software Loaders
- Utilities
- Operator Control/Monitor (Debug) Job

DATA STRUCTURES

- PR Configuration Table
- CPU Configuration Table
- CPU State Table
- System Job Index
- System Job List
- Job Dispatch Queues
- Job Clock Queues
- Packet Buffers

FIGURE 7 MAJOR ELEMENTS OF THE IPR OPERATING SYSTEM SOFTWARE

CONSOLE I/O AND INTERRUPT SERVICE - Provides local I/O channel console input/output services for the operating system and system jobs. Capabilities are provided for time sharing the single console between the several independent user processes.

RADIO/1822 DMA I/O INTERRUPT SERVICE - Queues and executes Radio and 1822 DMA I/O events as requested by the operating system and system jobs.

LOCAL CONSOLE/DOWN-LINE LOADERS - Implements the loading of the operating system and system jobs via the local console (RS-232) interface or remotely via the radio or 1822 DMA I/O interfaces.

BUFFER MANAGEMENT - Allocates the eleven packet buffers upon request from the operating system and system jobs.

PR/CPU CONFIGURATION TABLES - Contain data to define PR and CPU initialization.

CPU STATE TABLES - One for each CPU which contain dynamic data to define the execution priority of the CPU and define the CPU machine state in the event of failure.

JOB CLOCK QUEUE - Queue of jobs and time interval to reschedule job execution.

SYSTEM JOB LIST/JOB INDEX - Defines resident jobs and current execution status and machine state of resident system jobs. A system job may be: idle (never executed when called), halted (due to failure or operator command), checkpointed (execution stopped due to other system halt), suspended (waiting for next job call), or busy (executing).

JOB DISPATCH QUEUES - Defines call to execute system job plus call parameters to define reason for call. Three queues are implemented, one each for priority 1, priority 2, and priority 3 jobs.

3.2 Structure and Organization

The 32K word address space of each CPU is divided into 4 mutually exclusive zones. Each CPU implements one additional address bit for a total of 65K addresses which is used internally for byte addressing. Each zone is dedicated to specific functional elements of the system software as illustrated in Figure 8.

Each CPU implements a separate zone 1 which contains the CPU's interrupt and XOP vectors and workspaces used by the CPU for execution of the operating system.

CPU ADDRESS	CPU ADDRESS SPACE	ZONE	BUS ADDRESS SPACE	BUS ADDRESS
0000	256 WORDS RAM: -OPERATING SYSTEM WORKSPACES -INTERRUPT/XOP VECTORS	ZONE 1	ZONE 1 - CPU 1 RAM	00000
			ZONE 1 - CPU 2 RAM	00100
0200	10.5K WORDS RAM: -JOBS *CODE *WORKSPACE *LOCAL BUFFERS	ZONE 2	ZONE 2 BOTH CPUs RAM	00200
55FE				02BFF
8000	4K WORDS RAM: -GLOBAL DATA BASE -PACKET BUFFERS	ZONE 3	ZONE 3 BOTH CPUs RAM	04000
9FFE				04FFF
A800	3K WORDS RAM: -OPERATING SYSTEM *CODE *LOCAL BUFFERS	ZONE 4	ZONE 4 BOTH CPUs RAM	FD400
BFFF				FDFFF
D800	1K WORDS ADDRS: -I/O TAGS		ZONE 4 BOTH CPUs I/O TAGS	FEC00
E000	4K WORDS PROM: -OPERATING SYSTEM CODE		ZONE 4 BOTH CPUs PROM	FF000
FFFF				FFFFF

FIGURE 8 CPU ADDRESS MAPPING INTO IPR SYSTEM ADDRESS SPACE

Zone 2 is utilized for the system jobs' code space, workspaces, and local buffer with exception of the Debug job which resides in zone 4. Each job is relocatably loaded into zone 2 address space. The system loader assigns a bias register 2 value for each job equal to the beginning of each job. Jobs are packed in contiguous memory locations to optimize memory utilization.

Zone 3 is utilized for the system's global data base. It contains all buffer space which must be accessible by more than one job, such as the packet buffers.

Zone 4 contains the operating system code, tables, and time-varying buffer space. Additionally, zone 4 contains the peripheral device addresses.

All address mapping registers are defined by the operating system. System jobs perform no address mapping modifications. Address mapping is transparent to the system jobs.

3.3 Data Structures

Several tables and queues are defined by the operating system which are utilized to initialize, describe, control, and monitor job execution in the IPR. These are described below.

3.3.1 PR Configuration Table

The PR Configuration Table contains non-volatile information utilized to initialize the PR and define pertinent PR configuration data. This table contains table and queue addresses, initialization routine vectors, and images of the zone 1 CPU addresses, and other similar data. The PR Configuration Table is implemented in zone 4.

3.3.2 CPU Configuration Table

Each CPU is provided with a Configuration Table. This table contains non-volatile information utilized to initialize the CPU and to define pertinent configuration data unique to each CPU. The table includes: CPU identity data, CPU mapping register values, peripheral addresses (interval timer, interrupt register), table and queue addresses, and other similar data. The CPU Configuration Tables are implemented in zone 4.

3.3.3 CPU State Table

Each CPU is provided with a State Table. It contains volatile information to define the current execution state of the CPU. Each CPU State Table includes: a contention flag, the CPU name, current CPU execution priority, and the machine state of the CPU in the event of failure or termination of execution. The CPU execution priorities defined in order of highest to lowest priority are:

Executive
Job Priority 1
Job Priority 2
Job Priority 3 (Debug only)
Idle

Each CPU compares its current execution priority with the current execution priority of other CPUs to determine which CPU should process a common interrupt.

The CPU machine state is captured in the CPU State Table in the event of a CPU detected failure, if execution is terminated by operator command, if a software trace breakpoint is encountered, or if a job halts. The machine state is represented by the current value of the CPU registers (ST, PC, WP), and the contents of the workspace being used.

3.3.4 System Job/CPU Index

Each system job and CPU are uniquely defined by a name (ASCII character string) and an eight bit ID. The System Job Index is utilized by the operating system to define the existence of a CPU or to define a resident job. The table contains a one word entry for each system CPU or job. The CPU or job ID is utilized to index the table. A zero value entry defines a non-implemented CPU or a non-resident job. A non-zero entry defines the CPU State Table address of the CPU, or the address of the System Job List entry assigned to a resident job. A CPU's System Job Index entry is defined during PR initialization. A job's System Job Index entry is defined by the system loader when a job is loaded into the PR, except for the operating system Operator Control/Monitor (Debug) job and Station Ready Control (Flap) job which are defined during PR initialization.

3.3.5 System Job List

The System Job List consists of several variable length entries, each of which defines the current execution state of a resident system job. A System Job List entry is assigned to a job by the system loader when a job is loaded, except for the operating system Operator Control/Monitor (Debug) job and Station Ready Control (Flap) job which are assigned during PR initialization. A System Job List entry contains the following:

Contention Flag
Job Name
Job ID
Job Execution Priority
Job Initialization/Restart PC
Job Status (ST) Register
Job Workspace Pointer (WP) Register
Job Execution State
Executing/Last Executed CPU ID
CPU Execution Vector

Checksum Start Address
Checksum End Address
Checksum Datum
Job Dispatch Queue Address
Job Bias Register 2 Value
Job Relocation Displacement
Job Call Counter

The two byte contention flag equals 0 if the entry is empty, +1 if the entry is in use, or -1 if not in use.

The six byte job name contains the unique preassigned job name consisting of six ASCII characters terminated with ETX.

The one byte binary encoded job ID uniquely defines the system job. Job IDs are preassigned to all system jobs.

The one byte job execution priority equals binary 1, 2, or 3. One is the highest execution priority. Priority 3 applies only to the Debug job.

The two byte job initialization/restart PC defines the initial entry address of the job for power-up or restart initialization.

The job ST, PC, and WP entries, each two bytes in length, contain the current value of the job's CPU registers.

The two byte job execution state defines the current run status of the job. The most significant byte contains several one bit fields which describe the job halt status. Each bit, if equal to 1, defines the following:

Bit 0 - Watch Dog Timer Elapsed Fault
Bit 1 - Bus Response Timeout Fault
Bit 2 - Invalid Instruction Operation Code Fault
Bit 3 - Memory Parity Error Fault
Bit 4 - Software Trace Breakpoint Halt
Bit 5 - Unused Hardware Interrupt Fault
Bit 6 - Job Fault (XOP HALT, JBCTRL Call)

The least significant byte is binary encoded to describe the job's current run status, as follows:

VALUE (BASE 16)	RUN STATUS
00	Idle
01 - 03	Suspended
04	Halted
05	Checkpointed
07	Busy
08 - FF	Invalid (Halted)

A job is idle if it has never executed or is being reinitialized. A job suspends (ceases executing) via appropriate operating system call when it has no further processing to perform. It is re-executed when called by the operator, by another job, or when an event previously requested by the job completes, or a job specified time has elapsed. A job is checkpointed (execution ceases with the machine state captured) if the PR is halted due to failure or trace breakpoint. Job execution is halted due to failure or software trace as described above in bits 0-5. A job is busy when executing by one of the PR CPUs.

The one byte CPU execution vector defines the system CPUs which may execute the job. Each bit, if one, enables execution.

BIT	CPU
bit 7	CPU 1
bit 6	CPU 2
bit 5	CPU 3
bit 4	CPU 4
bit 3	CPU 5
bit 2	CPU 6
bit 1	CPU 7
bit 0	CPU 8

The two byte checksum start address is the address of the first word accumulated into the checksum calculation. The two byte checksum stop address is the address after the last word accumulated into the checksum calculation. The two byte checksum datum is the value of the final accumulation. The datum is generated by the system loader, and then used by the job's checksum test during Restart Initialization. If the checksum start address is zero, checksum generation/testing is not done.

The two byte job dispatch queue address inserted by the system loader defines the absolute job dispatch queue address used for the job calls.

The two byte job bias 2 register value defines the 256-byte zone 2 memory page where the job resides. This value is selected by the system loader.

The two byte relocation displacement, determined by the system loader, defines the job's displacement (in bytes) from the beginning of the zone 2 memory page.

The two byte job call counter contains the binary count of the number of calls residing on the Job Dispatch Queue for the job.

3.3.6 Job Dispatch Queues

The job dispatch queues contain calls for execution of the system jobs. Three queues are currently defined: one each for priority 1 jobs, priority 2 jobs, and priority 3 jobs (Debug only). Each CPU bids for the job dispatch queues and executes jobs defined therein. Several calls to the same job may exist at one time. Suspended jobs may be executed by either CPU, execution vector permitting. Checkpointed jobs may only be re-executed by the CPU which previously executed the job. A CPU will idle (cease execution with IDLE instruction) if no calls exist which may be executed. This is done to eliminate bus contention by a CPU which has no useful processing to perform.

Job calls are generated when a job is called by another job, when a previously job requested event has completed, or by operator command via the console.

Each job call consists of a three-word entry in the appropriate job dispatch queue. The first word defines the called job's System Job List address. The second word is the job call ID which consists of a one byte field defining the caller ID and a one byte event code. The caller's ID equals 0 for jobs called by the operating system. It defines the calling job's ID for interjob calls. The event code uniquely defines the purpose of the call. The third word is data, typically a buffer address. When a job is reinitiated from the point of suspension, the Job Call ID and data word are stored respectively in the job's registers R2 and R3.

3.3.7 Job Clock Queue

When a job suspends for time, an entry is made in the job clock queue. Each clock queue entry defines the job ID and the job specified time interval. When the specified time interval has elapsed, the job is called by the operating system. The entry is deleted when the job is called. A second suspend for time request deletes a previous entry for the job.

The clock queue contains a contention flag and the elapsed time clock value when the queue was last serviced. Both CPUs service the single Job Clock Queue.

3.3.8 Packet Buffers

The operating system implements eleven packet buffers in zone 3. Each packet buffer consists of 160 16-bit words as illustrated below:

User Space (20 words)
Packet Assign/Release Flag (1 word)
Packet Buffer Queue Chain Address (1 word)
Packet Preamble (3 words)
Packet Header, Text, CRC, 1822 DMA Status (135 words)

The User Space is provided for temporary buffer. It is further defined by the Radio/1822 DMA I/O routines. The Packet Assign/Release Flag specifies if the packet is assigned to a user. The Packet Queue Chain Address contains an absolute address pointing to the next packet buffer in the circular packet buffer queue. The Packet Preamble contains the three radio transmitted packet preamble words. One hundred thirty-five words are provided to buffer the actual packet header and text (up to 127 words), CRC checksum (2 words), and 1822 DMA status (6 words).

3.4 Processes

3.4.1 Initialization/BIT Diagnostics

Initialization does the BIT Diagnostics and initializes the peripheral hardware and the operating system. It consists of two major parts, power-up initialization and restart initialization, each of which consists of several sections. Power-up initialization has the following sections:

- PROM checksum test
- RAM test with incrementing test values
- RAM test with decrementing test values
- Mapping register read back test
- Mapping register address memory test
- RAM test/initialization with zero
- Bootstrap load and checksum generation/test

Restart initialization has the following sections:

- OS initialization and checksum tests
- OS initialization and jobs' checksum tests (and down-line load, if warranted)
- Error summary

In addition, the peripheral hardware is cleared, the 1822 READY line is set low, CPU interrupts are cleared, the mapping registers are initialized, and the zone 1 workspaces are initialized as a result of any stimulus that causes initialization.

The sections are executed in the order listed, with power-up initialization followed by restart initialization. Each section is executed sequentially by each CPU. Both CPUs finish execution of a section before either begins execution of the next. Both CPUs execute all of the sections except bootstrap load and checksum generation/test, which is executed by only the first CPU.

Power-up initialization is done as a result of one of the following:

- First applying power to the PR
- Pushing the INIT button
- Executing the IN console command
- An initialization error with auto-restart enabled
- An operational failure with auto-restart and down-line load enabled

Restart initialization is done following power-up initialization. After restart initialization has been completed, it is redone as a result of one of the following events:

- Restoring power after having been off
- Pushing the RSET button
- Executing the RS console command
- An operational failure with auto-restart enabled but down-line load disabled

If power is restored or the RSET button is pushed during the execution of initialization, the currently executing section is restarted.

3.4.1.1 <u>Description of Initialization Sections</u>
<u>Section 1: PROM checksum test</u>

Does checksum tests of the PROM portions of the operating system. Testing includes checking for the absence of memory parity errors and bus time-out errors while accessing the checksummed address spaces.

Section 2: RAM test with incrementing test values

Tests the RAM by executing the following:

1) Writes an incrementing pattern wherever RAM is detected from bus address 00000 up to the zone 4 boundary using the zone 1 bias register, followed by zone 4 up to the I/O address tags using the zone 4 bias register.

2) Reads back from the RAM in the same order and checks for the correct pattern and for the absence of memory parity and bus time-out errors.

Section 3: RAM test with decrementing test values

Tests the RAM by executing the following:

1) Writes a decrementing pattern (the inverse of that used in the incrementing RAM test) wherever RAM is detected from bus address 00000 up to the zone 4 boundary using the zone 1 bias register, followed by zone 4 up to the I/O address tags using the zone 4 bias register.

2) Reads from the RAM in the same order and checks for the correct pattern and for the absence of memory parity and bus time-out errors.

Section 4: Mapping register read back test

Tests the ability of the mapping registers to be loaded and read correctly by executing the following:

1) Writes an incrementing pattern, using the zone 1 bias register, at every 256-word bus address outside of zone 4.

2) For every 256-word bus address where there is RAM present, for the zones 3, 2, and 1 mapping registers, and for CPU addresses A600 and 5800, loads and reads back from each bias and limit register and checks that it reads back correctly.

Section 5: Mapping register address memory test

Tests the ability of the mapping registers to address memory correctly by executing the following:

1) Writes an incrementing pattern, using the zone 1 bias register, at every 256-word bus address outside of zone 4.

2) For every 256-word bus address where there is RAM present, for the zones 3, 2, and 1 mapping registers, and for CPU addresses A600 and 5800, loads the bias and limit registers appropriately, reads the memory location, and checks for the correct pattern.

Section 6: RAM test/initialization with zero

Tests/initializes the RAM by executing the following:

1) Writes zero wherever RAM is detected from bus address 00000 up to the zone 4 boundary using the zone 1 bias register, followed by zone 4 up to the I/O address tags using the zone 4 bias register.

2) Reads back from the RAM and checks for zero and for the absence of memory parity and bus time-out errors.

Section 7: Bootstrap load and checksum generation/test

Loads the non-resident portions of the non-volatile part of the operating system. Generates the checksum value for each newly-loaded portion of the operating system or does a checksum test if the checksum value was loaded. Testing includes checking for the absence of memory parity and bus time-out errors while accessing the checksummed address spaces.

Section 8: <u>OS initialization and tests</u>

Does a parity test of all the zone 4 RAM. Does checksum tests of all portions of the operating system. Testing includes checking for the absence of memory parity and bus time-out errors. Initializes the CPU state table, the job dispatch queues, and the clock queue, and installs the Debug job and Flap job. Initializes console I/O and other OS zone 4 flags and pointers.

Section 9: <u>OS initialization and jobs' checksum tests (and down-line load, if warranted)</u>

Does a parity test of all the zone 4 RAM. Initializes buffer management and DMA I/O flags and pointers. Installs the interrupt and XOP vectors. Sets the vector indicating the implemented CPUs and initializes the vector indicating the enabled CPUs. Installs the pointer to the CPU "job" entry in the system job index. Initializes the entries and does checksum tests of the resident jobs in the system job list. Checksum testing includes checking for the absence of memory parity and bus time-out errors while accessing the checksummed address spaces. If no jobs except Debug and Flap are resident, does a job down-line load, providing that the auto-restart and down-line load switches are set.

Section F: <u>Error Summary</u>

Displays the total number of errors during initialization in the front panel data LEDs, blinking the display if the number is non-zero.

3.4.1.2 LED Display and Error Logging

As each section is entered by each CPU, the section ID is displayed in the second character of the front panel code LEDs, the CPU number is displayed in the first character of the front panel data LEDs, and zeros are displayed in the other characters. The exception is Section F, in which the total number of errors is displayed in all 4 characters of the data LEDs.

Every error detected in the diagnostic tests causes an error code to be displayed in the last 3 characters of the front panel data LEDs and causes the whole display to be blinked. After an error has been displayed for 2 seconds, execution resumes unless certain hardware switches are set. If the auto-restart switch is set, power-up initialization is redone. The exception is when there has been a load error, in which case the down-line load switch, as well as the auto-restart switch, must be set in order for power-up initialization to be redone; if both switches are not set, execution stops after a load error. Special initialization mode switches cause special sequencing for debug purposes.

3.4.2 Job Control

Several processes are utilized to schedule, monitor, and control job execution in the IPR. These are listed below:

- Job Call
- Job Scheduler
- Interval Timer/Clock Queue
- CPU Directed Interrupt
- Failure Interrupt and Unused XOP/Interrupt

3.4.3 Job Call

The job call process is invoked via user XOP or operating system BLWP call. It generates a job call on the appropriate job dispatch queue and generates a CPU directed interrupt to all enabled CPUs to define the existence of a new system processing task.

Calls to execute a system job are generated by the operating system as a result of the following:

- Completion of an event previously requested by the job. These include Radio/1822 DMA I/O, console input, and suspension for time interval elapsed.
- Upon request from another system job.
- Upon request from the operator via the local console command XJ (calls all idle jobs).
- Upon auto-restart (calls all resident jobs).

A job call is defined by a three-word entry in the job Dispatch Queue corresponding to the job execution priority. These words consist of the job's System Job List address, Job Call ID, and data word. Several calls of the same job may exist at one time. The Job Dispatch Queue data structure provides the service of message sending as well as job scheduling.

3.4.3.1 <u>Job Scheduler</u>

This process is invoked by user process XOP or operating system BLWP call. Its task is to save/restore the machine state of the system jobs and initiate jobs defined in the job dispatch queues.

Each CPU independently examines the Job Dispatch Queues to determine if jobs are to be executed. The queues are examined in order of priority (1,2,...). Job calls in each queue are serviced first in/first out. Preempted (checkpointed) jobs are re-executed only by the CPU which last executed the job. Suspended jobs are re-executed by either CPU. Idle jobs (jobs which have never executed, or are being initialized) are initiated at the Initialization/Restart PC. Suspended or checkpointed jobs are always reinitiated at the point execution ceased as defined by the job's current ST, PC, WP register values maintained on the System Job List.

If no job calls exist which may be serviced, a CPU will go idle (execute IDLE instruction). In the idle state no bus accesses are made by the CPU which eliminates bus contention by a CPU with no useful processing to perform. A CPU leaves the idle state upon the occurrence of any hardware interrupt.

A CPU deletes the Job Dispatch Queue entry, restores the job's machine state and initiates the job. The job call ID and data word are copied from the Job Dispatch Queue into a suspended job's workspace register R2 and R3.

This process is called by a system job to relinquish control to the operating system upon completion of a processing task. The job may:

- Suspend
- Suspend for Time
- Suspend for Time Conditional
- Halt (Fault)
- Checkpoint

Suspend ceases job execution until recalled. Suspend for time ceases job execution until the job specified time interval elapses or another call occurs. Suspend for time conditional is suspend if calls exist for the job on the Job Dispatch queue, otherwise suspend for time. Halt is utilized when the job detects a failure which prohibits continued PR operation. Auto-restart is invoked if selected via hardware switch or IPR execution is halted. Checkpoint is used by the operating system to stop job execution when the PR is halted. It generates a job call which is placed at the top of the Job Dispatch Queue.

3.4.3.2 Interval Timer/Clock Queue

This process is invoked from the job scheduler or via interval timer interrupt. It maintains the Job Clock Queue which defines the job and time intervals remaining before job recall. The job call process is invoked for jobs with elapsed time intervals. The queue maintains the last call interval for each system job utilizing this capability.

3.4.3.3 CPU Directed Interrupt

This routine is invoked via interrupt generated by either CPU. The interrupt is generated by the job call process or when the PR is being halted. If the PR is halted (stand-alone mode), a busy job is checkpointed and/or the CPU goes idle. In normal operation an idle CPU enters the job scheduler. A CPU executing a job disregards the interrupt.

3.4.3.4 Failure Interrupt/Unused Interrupt or XOP

This process is invoked via failure interrupt. The interrupt is the result of a bus timeout, watch dog timer elapsed, invalid instruction code, or memory parity error failure.

All unused interrupt and XOP vectors invoke the failure interrupt process. The job scheduler process is invoked to halt the failed PR.

If a failure reoccurs before the Operator Control/Monitor (Debug) job may report the failure the CPU is idled with interrupts disabled. Failure is identified by front panel LED display.

3.4.4 Console I/O

This process serves as the interface between jobs and the local (RS-232 I/O Channel) console. Services provided to the system jobs are listed below:

- Assign/Release Console
- Output ASCII Message
- Output(binary) Data as ASCII Message
- Request ASCII Message Input
- Release Input Buffer

The assign/release console is used to assign the console output to the requesting job for one or more logically connected outputs and subsequently release the console output for other jobs. Assign console timeout is provided to prevent lockout.

Output ASCII message outputs an ASCII character string terminated by ETX (03).

Output binary data encodes any number of 4 bit binary groups in ASCII hexadecimal (0-9, A-F) and outputs the string preceded and followed by a single space.

Input message requests are queued. Operator input directed to a job requesting input is buffered. When the input is complete the requesting job is called and is provided with the input buffer address. The job releases the input buffer after processing the operator input.

Output is buffered by the console I/O process. Output from different jobs is separated by output of the job name by the console I/O.

The console I/O routines are invoked via use of the XOP instruction. The calling job passes an event code, and possibly an additional parameter, in its own workspace. The linkage provided by XOP allows the I/O routines to non-destructively read those parameters from the calling job's workspace.

Console I/O is interrupt driven in normal operation. The I/O interface is polled when the PR is halted (stand-alone mode). When halted the Operator Control/Monitor job utilizes a special console I/O interface which preempts other job use of the console.

When a console interrupt occurs, both CPUs (if their interrupts are enabled) enter the interrupt handling routine and compare their execution priority with the other CPU's execution priority. If a CPU's priority is less than, or equal to, the other CPU's priority, this CPU will immediately bid for the interrupt contention flag. If the flag is found negative, the CPU clears the global and local interrupts, switches to a common zone 4 workspace, and services the interrupt. If the contention flag is found positive, i.e., the other CPU is already servicing the interrupt, the interrupt routine is exited. If a CPU's priority is higher than the other CPU's priority, this CPU will delay 20 usec before bidding for the contention flag.

3.4.5 Radio/1822 DMA I/O

The DMA I/O routines provide the following services for Radio and 1822 receive (RX)/transmit (TX):

- Initiate a DMA I/O event
- Service completion of a DMA I/O event
- Idle a previously requested DMA I/O event

3.4.5.1 Radio/1822 DMA I/O Initiate

Whenever a DMA I/O initiate request is issued, one of the following operations is executed:

1. If the DMA channel is not currently busy with a previous I/O request, then the DMA request is initiated.
2. If the DMA channel is busy, then the I/O request is queued and will be initiated as soon as the channel is free.

Using the information passed by the user in the DMA Packet Buffer Overhead, the DMA I/O event is initiated by loading the DMA hardware Word Count, Address, and Control Registers.

The Radio RX control parameters enable the user to enable the radio receiver hardware in the following modes of operation:

- Reception of packets through either of the two Radio RX DMA channels.
- Specifically specify one of the two RADIO RX DMA channels for reception of packets.
- Loop test of the receiver interface with and without the channel selector.

The Radio TX control parameters enable the user to transmit packets with the following options:

- ALOHA with and without randomization.
- DISCIPLINED ALOHA with and without randomization.
- Carrier Sense with and without randomization.

. Radio turnaround with the receiver interface.

. Radio turnaround with the receiver interface and channel selector.

The 1822 RX and 1822 TX control parameters allow the user to select the following modes:

. External reception or transmission of packets over the 1822 interface.

. Loop test of the 1822 receiver and transmitter.

3.4.5.2 Radio/1822 DMA I/O Service Completion

Completion of a DMA I/O event is indicated by generation of a DMA hardware interrupt. Completion service includes clearing the DMA hardware interrupt and inputing contents of the DMA hardware Word Count, Address and Status registers into the DMA Packet Buffer Overhead. At this point the DMA I/O queue is reviewed for further I/O initiate events.

3.4.5.3 Radio/1822 DMA I/O Idle

A DMA I/O event can be idled any time it has been queued, initiated or service completed.

3.4.6 Local/Down-Line Load

These processes provide the capability to load and load verify relocatable and absolute origin TI 9900 object code and data into the IPR. The object may be input via the console I/O channel interface (normaly from TI Silent 700 tape cassettes) or via the IPR radio or 1822 DMA I/O interfaces.

The loader accepts standard TI 9900 assembler object format ASCII encoded records with the addition of two characters. These are the " < " ($3C_{16}$) character immediately preceding the first record of a file, and a "$" ($2A_{16}$) is appended to the end of a load file by the network load service to delimit the down-line load object file via radio or 1822.

The first origin of the program indicates whether the program is absolute or relocatable. If the first origin is non-zero, the program is assumed to be absolute. If the first origin is zero, the program is assumed to be a relocatable job. If the program is a relocatable job, the loader will check the system job index to determine if the job has previously been loaded. If so, the incoming program will be loaded over the previous version. Care should be taken that, if the new version of an existing job is longer, the system job list must be

cleared with a Terminate Job (TJ) command before loading. If the job does not already exist, the loader scans the system job list until it finds an empty entry. Any absolute origin data in the relocatable job will be relocated into the system job list entry. It is not necesary that all jobs be loaded at the same time.

Relocatable jobs are loaded into zone 2 and are placed in consecutive memory locations. Absolute programs may be loaded anywhere. Load verification is automatically performed by writing the data into memory and immediately reading it back to ensure its proper storage.

Load verification may be performed on absolute or relocatable programs. If relocatable, the loader obtains the bias and relocation displacement for that job from its system job list entry. The loader does not verify the contents of a job's system job list entry.

The loader is invoked by the operating system if auto-restart and down-line hardware straps are enabled or by the Operator Control/Monitor job as a result of an operator command. Loading is permitted only in the PR halted (stand-alone) mode.

3.4.6.1 Local Console Load

The local console loader accepts input from the PR I/O channel interface. The interface is polled for input as opposed to normal interrupt driven operation. Operator actions and load errors are defined by messages output to the console.

3.4.6.2 Down-line Load

The down-line load accepts the ASCII encoded object contained in packets received via the 1822 and radio DMA I/O interfaces. The request to be down-line loaded is defined by the Load Request ROP packet periodically transmitted by the PR.

The IPR transmits the Load Request ROP every five seconds alternately via 1822 and radio. The Load Request ROP is transmitted at full power, 100 Kbit rate, carrier sense channel access, at the RF frequency specified in the IPR hardware switches. Once the load begins Load Request ROP transmissions are suspended until 5 seconds has elapsed since the receipt of the last valid Load Data Packet.

The Load Data Packet may be received via 1822 or 100/400 Kbit radio. To be accepted the packet must be received correctly, be of the Load Data type, match the load data file name requested and be correctly sequenced. The IPR verifies the received file name against what is stored and, if it matches, stores the name received. The previous name stored (or requested) may be a shortened version, such as the default case ("ETX"). Subsequent packets must match the earlier ones in file name. Additional checks are made on the load data. If load data text errors are detected the down-line load is aborted with appropriate local error display. The format of the Load Request ROP packet is shown in Figure 9.

Once loading is started, the IPR will only accept Load Data Packets which are from the same source in addition to having the expected file name and sequence number. If an acceptable Load Data Packet is not received for 5 seconds, the IPR will begin sending Load Request ROPs and will accept load commencement from any source as long as the conditions of file name and sequence number are met. If no acceptable Load Data Packet is received for 15 consecutive Load Request ROPs (75 seconds), the IPR will declare a Load Error 3 and either re-initialize or halt, depending on the states of the auto-restart and down-line hardware straps. In either case an error message is output if a console is attached.

The load data file name is included in the Load Request ROP for two reasons: One, the IPR will indicate the request for the IPR OS boot load and network protocol load as separate load requests. Two, the operator may request via local console command the loading of off-line diagnostic programs as well as protocol. This allows a simpler console device (keyboard, printer, no load capability) to be used for field maintenance.

The error data provides a mechanism for the return of diagnostic test results to the Network Control Center (station). This data may be generated by IPR BIT diagnostics and by the remote execution of the off-line diagnostics. The goal is to provide the capability for an operator at the NCC (station) to execute the off-line diagnostic programs in a remote network IPR and obtain the summary test results.

The checksum is provided for a hardware/software verification which is performed by the protocol software.

The load object is contained in Load Data Command Packets. This packet format is shown in Figure 10.

Load data ASCII text must be packed in the order left byte, right byte, in consecutive Load Data packet text words. If the load data is an odd number of bytes the last byte should contain a NULL (00) character. Load data records may be fragmented into more than one Load Data packet. The load data format is identical to that loaded locally via tape cassette except that a down-line load must be terminated by the character "$" ($24_{16}$). The load service will need to append this character to the load data files supplied by Rockwell-Collins.

WORD	DEFINITION
1	Hop count, header length, packet length Bits 0- 3 => Hop count = 0 Bits 4- 7 => Header length = B Bits 8-15 => Packet length
2-11	Unused header words all set to zeros.
12	IPR ID from hardware switches
13	LROP flags Bit 0 => Label bit Bit 1 => Distress LROP Bit 2 => Load Request bit Bits 3-4 => PR Type 00 - UPR 01 - IPR 10 - EPR 11 - LPR This word always set to 2800_{16}.
14	Next expected Load Data Packet sequence number. Incrementing count with initial value of 0000.
15-18	Load data file name. ASCII character string terminated by ETX (03). Maximum length 8 characters. ETX only defines request for latest version of network protocol. The IPR OS boot load file name is "IPROS4 ETX".
19	Left byte defines IPR job ID which generated error data below. Right byte is binary count defining length of error data in 16 bit words.
As required	Error data.
Last word	Checksum.

FIGURE 9 DOWN-LINE LOAD REQUEST ROP PACKET FORMAT

WORD	DEFINITION
Header	The IPR down-line loader uses the following fields of the packet header: 1. Bits 4-7 of the first header word is the packet header length and is used to locate the start of the packet text. 2. Bits 8-15 of the first header word is the packet length and is used to determine the length of the load data. 3. Bit 11 of word 5 is the active acknowledge (ACT) bit and is set when transmitting an active acknowledgement on the radio.
Text Word 1	The upper byte of the first text word indicates the type of command packet. The PR Load Data command is designated by a 30_{16} in the upper byte of the first text word. The lower byte of this word is ignored.
Text Word 2	This word contains the ID of the PR to be loaded.
Text Word 3	This word contains the Load Data packet's sequence number. The sequence number of the first packet in the load is 0000, and each subsequent load data packet has a sequence number one higher than the last packet's number.
Text Word 4	Beginning in this word is the name of the file being loaded. The name is an ASCII character string terminated by an ETX. The entire string (including the ETX) must be less than, or equal to, eight characters in length.
Remainder of text	Immediately following the file name's ETX, the IPR's down-load routines begin looking for the start of the ASCII load data. Null (00) bytes are ignored by the loader.

FIGURE 10 DOWN-LINE LOAD DATA COMMAND PACKET FORMAT

The load data file name must be contained in every Load Data Packet. This name is compared with the name in the Load Request ROP. They must be equal for the down-line loader to accept the packet. However,if the Load Request ROP load data file name is ETX only, the loader will accept any load data file. This feature is provided to permit a remote operator to load and execute any file (off-line diagnostics or protocol).

3.4.7 Buffer Management

The operating system provides eleven packet buffers contained in Zone 3 memory. Each packet buffer (described in Section 3.3.8) contains an overhead area for system software use, the packet preamble for radio transmissions and space for the maximum length packet (header, text, CRC).

The packet buffers are constructed when the system is initialized. Packet buffers are allocated to users upon request and are released by users when no longer needed. Packet buffers are assigned and released via appropriate call to the operating system utilizing the extended operation (XOP) instruction.

3.4.8 Utilities

Four utility processes utilized by the jobs are provided. They are invoked by user job XOP.

One process updates the vector indicating the enabled CPUs and, in doing so, disables operation of all but the selected CPU. This enables operation of a single CPU for testing and maintenance.

The second process provides the absolute address of the CPU Configuration Table to permit job access to the data contained therein.

The third process buffers error data from a job or operating system process which is subsequently included in Load Request ROP packets.

The fourth process displays or alters locations in memory to permit remote AM/DM commands via Command Packets

3.4.9 I/O TAG Read/Write Service

This process provides an interface between the user jobs and various hardware functions. This process is invoked by user job XOP.

The following seven functions are provided:

Read MS word of elapsed timer.
Read LS word of elapsed timer.
Read both words of elapsed timer.
Reset elapsed timer.

Read PR ID straps latch.
Read RAM image of MODE straps latch.
Write to front panel CODE and DATA LEDS.

3.4.10 Operator Control/Monitor (DEBUG) Job

The operating system provides system monitoring and control capabilities to an operator via command input and message display at the local console. These capabilities are utilized to monitor and control normal execution of the system jobs and provide necessary tools to support system software/hardware debugging.

The Operator Control/Monitor Job is executed as any other system job in normal operation of the PR. It executes in a single CPU when the PR is halted (stand-alone mode). It is invoked upon completion of PR initialization, when the PR fails or a software trace breakpoint is encountered. It may be invoked by operator console command or by another system job.

The functional capabilities provided which are selectable via operator or job command are listed below:

- Terminate Jobs (TJ)
- Display Memory or Peripheral Interface (DM)
- Alter Memory or Peripheral Interface (AM)
- Display Job Registers (DR)
- Invoke Restart Initialization (RS)
- Invoke Power-Up Initialization (IN)
- Halt Jobs (HJ)
- Display Job Status (DJ)
- Execute Operator Routine (GO)
- Initiate Execution of Jobs (XJ)
- Alter Job Registers (AR)
- Set Software Trace Breakpoints (ST)
- Clear Software Trace Breakpoints (CT)
- Down-Line Load (DL)
- Down-Line Load Verify (DV)

. Console Load (LD)

. Console Load Verify (LV)

3.5 Operational Description

3.5.1 Initialization

Power-Up initialization is invoked by depression of the INIT pushbutton, by operator console IN command, or by PR failure which warrants power-up initialization. Both CPUs participate in the initialization which consists of several BIT diagnostic tests, loading of the remaining portions of the operating system, and initialization of the operating system RAM. Execution of each segment of initialization is shown in the front panel LED display. A blinking display identifies failure. If auto-restart is enabled, a failure will re-invoke power-up initialization. Upon completion of power-up initialization, restart initialization is invoked.

Restart initialization is invoked by depression of the RSET pushbutton, by operator console RS command, by PR operational failure if just auto-restart is selected, and upon completion of power-up initialization. It parity checks the RAM used by the operating system. It checksum verifies operating system and resident job code space. It initializes dynamic OS RAM to an idle (never executed) state. If no jobs exist, a down-line load is requested if auto-restart and down-line load are enabled. Execution of each segment of initialization is shown in the front panel LED display. A blinking display identifies failure. If auto-restart is enabled, a failure will invoke power-up initialization. Upon completion of restart initialization the Operator Control/Monitor (Debug) job is executed by one CPU.

3.5.2 Stand-Alone (Halted) Operation

Upon completion of initialization the Operator Control/Monitor job is initiated in a single CPU in stand-alone (PR halted) mode of operation. PR halted mode is also invoked when a PR fails, a job halts, a software trace breakpoint is encountered, and by operator IN, RS, HJ, and TJ console commands, and depression of the front panel INIT and RSET pushbuttons.

Stand-Alone (halted) is a special mode of operation for operation in a degraded PR for testing and maintenance, and as a transition state before normal operation is invoked. In stand-alone mode, the Debug job executes independently of the normal scheduler in a single CPU with all but failure interrupts inhibited. The other CPU is idle at interrupt level 3. All other job execution is inhibited. Console I/O and DMA I/O (for down-line loading) are used in a polling non-interrupt driven mode. Debug execution is independent of any

previously set contention flags or zone 1 contents or address mapping. All console operator input is directed to Debug job via a dedicated input buffer. Console output preempts and is independent of any previously queued console output. All operator console commands are valid.

3.5.3 Normal Operation

Normal operation is invoked by the Debug job automatically or via operator command. If auto-restart is enabled and jobs are resident Debug calls all idle or halted jobs (auto XJ command). A CPU directed interrupt is generated to all enabled CPUs. Normal operation is invoked by operator XJ console command.

In normal operation both CPUs independently execute the operating system and scheduled system jobs. Operation is event driven via the normal scheduler. Debug operates as any other system job. Some Debug operator commands are not valid.

4 JOB DESCRIPTION

The applications processes of the PR are called jobs. The PR operating system software provides for loading, scheduling, execution, and on-line debugging of the system jobs.

The network protocol processes, off-line diagnostic processes, and operator control/monitor (debug) process are executed as system jobs.

4.1 Job Structure

A job is partitioned into three major sections: The System Job List data, Volatile Local Buffer Space, and Non-Volatile Instruction Code Space. This is preceded by descriptive comments on the assembly listing. Volatile and non-volatile memory space are separated so that instruction space may be placed in read only memory (ROM). This structure in assembler syntax is illustrated in Figure 11.

The job's System Job List is the data which follows the absolute origin statement (AORG >0). Absolute data must be defined as shown. The remaining data is defined as follows. Each job is assigned a unique five ASCII character job name, and a unique binary encoded job ID. All jobs are assigned a job execution priority. At present two are implemented; priority 1 the highest, and priority 2 the lowest priority. The job initialization/restart PC defines the job entry PC for initial execution of the job. All jobs are in memory restartable. Jobs are required to load (LWPI instruction) the workspace pointer upon initialization. The jobs ST, PC, WP CPU registers define normally the job's initial status ($00F_{16}$), initialization/restart PC and initial workspace pointer. The CPU execution vector defines which PR CPUs may execute the job. Each bit equal to one defines a CPU which may execute the job. The bits numbered 0-7 left-right define respectively CPUs numbered 8-1. Because the IPR's dual CPUs are numbered 1 and 2, the normal value for this parameter is 03.

The beginning of the job address space is designated by the relocatable origin (RORG >200) statement. Jobs are relocatably loaded into the Zone 2 address space. Jobs are packed contiguously into the Zone 2 address space by appropriate selection of a bias register 2 and relocation displacement for each job.

```
        TITL    'job title'
        IDT     'job name'
        OPTION  XREF,OBJ
*
* Description of Job's Purpose and Scope
* Description of Job Call Event Codes
*    (if any) defined by Job
*
* System Job List Data
*
        AORG    >0
        DATA    >FFFF           Contention Flag
        TEXT    'job name'      5 Character ASCII Name
        BYTE    >03             ASCII Name Delimiter
        BYTE    >job ID         Job ID
        BYTE    >priority       Job Execution Priority
        DATA    label           Job Initialization/Restart PC
        DATA    >n              Job CPU ST Register
        DATA    label           Job CPU PC Register
        DATA    label           Job CPU WP Register
        DATA    >0              Job Execution State (Idle)
        BYTE    >0              Executing CPU ID (none)
        BYTE    >vector         CPU Execution Vector
        DATA    label           Job Checksum Start Address
        DATA    label           Job Checksum Stop Address
        DATA    0               Job Checksum Datum
*
*       Beginning of Volatile Memory Partition
*
        RORG    >200            Zone 2 Limit
*
*       (Volatile data buffers)
*
*       Beginning of Non-Volatile Memory
*       Partition (Non-Volatile instructions/data)
*
        END
```

FIGURE 11 JOB STRUCTURE IN ASSEMBLY LANGUAGE SYNTAX

4.2 Job Loading and Initialization

Jobs are created in the IPR by loading via the local I/O channel (normally from magnetic tape cassette) or by down-line loading via the 1822 or radio channels.

Jobs are relocatably loaded into Zone 2. The loader selects a System Job List entry and places its address in the System Job Index. The System Job List entry is initialized by data in the job load and by the loader. The initial execution state of the job is idle.

4.3 Job Execution

Initial job execution is invoked automatically or via operator console command by the Operator Control/Monitor job. Job execution begins at the initialization/restart PC. Jobs are required to initialize all local (internal) buffer space appropriate to initial execution of the job. External initialization is performed by the operating system.

Jobs execute to perform their defined tasks. They may utilize the resources of the operating system, call or be called by other system jobs. When current processing tasks are completed jobs suspend to await subsequent calls. Execution resumes at the point following suspension. Jobs may be non-concurrently executed by several CPUs as determined by the job's CPU execution vector. Jobs may execute prior to suspension up to the watch dog timer internal (approximately 2 seconds). If the time interval elapses during job execution, a system failure is declared.

4.4 Job Interfaces to the Operating System

System jobs interface with the operating system for the purposes of controlling the job's execution, to execute processes provided by the operating system, and to communicate with other system jobs. These interfaces are implemented with the extended operation (XOP) instruction. Each CPU implements sixteen XOP vectors in its Zone 1 memory space. Each two word vector consists of a workspace address, followed by a program counter (entry address) value. Each CPU utilizes a unique workspace (in Zone 1) for execution of the operating system process (in Zone 4). Contention is resolved with the ABS instruction operating on process contention flags.

The general format of the XOP call to the operation system is shown below:

```
XOP (Call ID), Vector Name
Return
```

The effective address specified by the first XOP instruction operand defines the location of the Call ID. The Call ID consists of one byte process ID (most significant byte), and one byte call event code (least significant byte). The process ID equals 00 for operating

system events. It equals the called job's ID for interjob calls. Each operating system event is assigned a unique event code which defines the desired event. The called job defines the event codes for interjob communication.

The vector name specifies one of the sixteen XOP vectors. Current vector assignments and associated event code values are shown below.

Additional linkage information is required for some calls. This is contained in the calling job's workspace registers.

One or more returns from the XOP call may be defined. The requested event may be processed immediately, in some cases it may be queued for later processing, or the XOP may specify the call of another job.

In some cases, a requested event when completed generates a recall of the job which originally requested the event. A job when recalled from the point of suspension determines the reason for the call by decoding data returned to its workspace registers R2 and R3.

4.4.1 Job Control

The job control calls are utilized by the system jobs to temporarily terminate the job's execution. Five calls are provided.

- Suspend for Time Conditional (SPNDTC)
- Checkpoint (CKPT)
- Suspend (SSPND)
- Suspend for Time Interval (SSPNDT)
- Halt (HALT)

The XOP call arguments are shown below.

```
JBCTRL   EQU      0
CKPT     DATA     >0005
SSPND    DATA     >0002
SSPNDT   DATA     >0003
HALT     DATA     >0004
SPNDTC   DATA     >0001
```

4.4.1.1 Job Checkpoint

This call is utilized by the operating system to preempt execution of a job. A system job may utilize this call to temporarily terminate execution (watch dog timer is reset). The job state upon reinitiation is identical to the job state prior to checkpoint. Job call event codes are not returned to the job's workspace. The operating system generates a call for a checkpointed job. The call is illustrated below:

```
XOP @CKPT,JBCTRL
Return - Job Reinitiated
```

4.4.1.2 Job Suspend

This call is used by the system jobs to temporarily suspend execution until recalled. The job is recalled at the point of suspension with the job call ID and data word stored in the job's workspace registers R2 and R3.

The job call ID consists of the calling job's ID or the operating system ID (00) in bits 0-7 and the call event code in bits 8-15 of R2. For job requested events, the job call ID (DMA I/O, terminal I/O etc.) equals the original XOP call ID value. For interjob calls it equals a job defined event code and calling job ID.

A system job, upon reinitiation from suspension decodes the event code to determine the purpose of the call. The data word (R3) further defines the call. R3 is typically a buffer address.

The call is illustrated below:

```
XOP @SSPND,JBCTRL
RETURN - JOB RECALLED WITH R2, R3 CONTAINING
         CALL ID AND DATA
```

4.4.1.3 Job Suspend for Time Interval

This call provides the capability for a job to reschedule its execution when the job specified time interval has elapsed. The job is recalled when the time has elapsed or if any other job call occurs. Job call due to time interval elapse is identified by the SSPNDT call ID (value 0003) returned in the job's workspace register R2. Re-execution of this call will negate a previous suspend for time call, unless already queued on the Job Dispatch Queue.

The call time interval is specified by the content of the job's workspace register R3. The interval is specified by a binary count in bits 1-15 of R3. R3 bit 0 (most significant bit) defines a scaling factor. If bit 0 equals 0, each count equals 1.667 milliseconds. If bit 0 equals 1, each count equals 426 milliseconds.

The call is illustrated below:

```
R3 = TIME INTERVAL
XOP  @SSPNDT,JBCTRL
RETURN - JOB RECALLED WITH R2 CONTAINING THE CALL ID
         AND R3 CONTAINING DATA
```

4.4.1.4 Job Suspend for Time Conditional

Operation is equivalent to suspend or suspend for Time Interval. Operation is suspend if the job is scheduled for execution defined by calls on the job dispatch queue. Operation is suspend for Time Interval if the job execution is not scheduled. This call is utilized to eliminate unnecessary job calls due to time interval processing when the job is already scheduled for execution.

The call is illustrated below:

```
R3 = TIME INTERVAL
XOP  @SPNDTC,JBCTRL
RETURN - JOB RECALLED WITH R2 CONTAINING THE CALL ID
         AND R3 CONTAINING DATA
```

4.4.1.5 Job Halt

This call is used whenever a job detects a failure or abnormal condition which precludes continued execution of the job. The operating system will halt system job execution whenever a failure is detected. These include: watch dog timer elapsed, memory parity error, bus response timeout, invalid instruction operation code, and software trace breakpoint trap.

A job halt (or failure) will invoke auto-restart initialization and the operator will be notified via local console messages with job execution halted until restarted by the operator depending upon the auto-restart hardware strap.

This call is illustrated below:

```
XOP @HALT,JBCTRL
```

4.4.2 CONSOLE I/O

Six XOP calls are provided. These are:

- Assign Console (ASGTRM)
- Release Console (RLSTRM)
- Release Input Buffer (RLSIB)

- Output ASCII Message (MSGO)
- Output Data as ASCII (DATAO)
- Request ASCII Message Input (MSGI)

The XOP vector name and event code values are shown below:

```
TRMIO    EQU       4
ASGTRM   DATA   >0040
RLSTRM   DATA   >0041
RLSIB    DATA   >0042
MSGO     DATA   >0043
DATAO    DATA   >0044
MSGI     DATA   >0045
```

4.4.2.1 Assign Console

This routine dedicates the console to the requesting job. Assignment should be used if several logically connected output requests will be made and intervening outputs by other jobs are not desired. The console will remain in the assigned state until the assigning job releases the console with a call to RLSTRM, or until the assignment times out, i.e., the assigning job has not made any output request for a certain period of time and another job has requested to use the console.

ASGTRM has two returns. The first return indicates the console is already assigned to another job. The second return indicates the assignment was performed.

To assign the console:

```
XOP @ASGTRM,TRMIO
RETURN   - CONSOLE ALREADY ASSIGNED TO ANOTHER JOB
RETURN+4 - CONSOLE ASSIGNMENT PERFORMED
```

4.4.2.2 Release Console

The routine is used to release an assigned console. Release will only be performed for the job to which the console is assigned.

To release the console:

```
XOP @RLSTRM,TRMIO
RETURN - CONSOLE RELEASED OR NO ACTION
```

4.4.2.3 Release Input Buffer

This routine releases an input buffer containing characters previously input from the console. Console input is stored in a buffer in zone 4 and the address of the buffer is given to the job receiving input. After using the contents of the buffer, the job must release the buffer for future console input. To release an input buffer, an address is passed in register R3 of the calling job's workspace. That address must refer to some location within the input buffer. The address passed need not be the starting address of the buffer.

To release an input buffer:

```
R3 = ADDRESS WITHIN THE INPUT BUFFER
XOP @RLSIB,TRMIO
RETURN - INPUT BUFFER RELEASED
```

4.4.2.4 Output ASCII Message

This routine outputs an ASCII character string to the console. The character string must be stored sequentially in memory in ascending addresses. The last character in the string must be an ASCII ETX (03). The message may begin or end on an even or odd byte boundary.

The calling job passes the address of the start of the message in register R3. MSGO moves the character string into a transmit buffer in zone 4. The I/O routines saves the ID of the last job to have a message placed in the transmit buffer. MSGO places the message "JOB XYZ" (where XYZ is the name of the calling job) in the transmit buffer between messages from different jobs.

MSGO has two returns. The first return indicates the inability to move the message into the transmit buffer. This can result from the console being assigned to another job or from the transmit buffer having insufficient space to hold the entire message. Upon successful return from MSGO, the calling job may reuse its own message buffer.

To output a message:

```
R3 = STARTING ADDRESS OF MESSAGE CHARACTER STRING
XOP @MSGO,TRMIO
RETURN   - UNABLE TO ACCEPT OUTPUT
RETURN+4 - OUTPUT ACCEPTED
```

4.4.2.5 Output Data as ASCII Message

This routine encodes a specified number of consecutive four bit binary numbers as hexadecimal ASCII characters and outputs the characters to the console. The number may begin or end on even or odd byte boundaries. The nybbles (nybble = 4 bits = half a byte) of the

data must be stored sequentially in memory with increasingly significant bytes being stored in decreasing addresses. Two parameters must be passed to DATAO. Register R3 must contain the address of the most significant byte of the number. Register R2 must contain the number of nybbles in the number. The nybble count may be any nonzero value. If the nybble count is odd, the most significant nybble must be right justified in the byte.

For example, suppose it is desired to output the number $F123B_{16}$, which is stored in locations 0801_{16}, 0802_{16}, and 0803_{16}. The number would be stored as follows:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Memory	0800	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	1
Address	0802	0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1

Note that the number is right justified and leading zeros are not required. The byte at location 0800 could be used to store all, or the least significant portion of another number. The job wishing to output this number would call DATAO with the address 0801 in register R3 and the nybble count of 5 in register R2.

DATAO encodes the data into hexadecimal ASCII characters and moves the characters into a transmit buffer in zone 4. DATAO also outputs a space immediately preceding and following the number. The I/O routines save the ID of the last job to have a message placed in the transmit buffer. DATAO places the message "JOB XYZ" (where XYZ is the name of the calling job) in the transmit buffer between messages from different jobs.

DATAO has two returns. The first return indicates the inability to move the decoded data into the transmit buffer. This can result from the console being assigned to another job or from the transmit buffer having insufficient space to hold the encoded data. Upon successful return from DATAO, the calling job may reuse or change the numbers.

To output data as ASCII characters:

```
R3 = ADDRESS OF NUMBER'S MOST SIGNIFICANT BYTE
R2 = NYBBLE COUNT
XOP @DATAO,TRMIO
RETURN   - UNABLE TO ACCEPT OUTPUT
RETURN+4 - OUTPUT ACCEPTED
```

4.4.2.6 Request ASCII Message Input

This call indicates the willingness of the calling job to accept input from the console. MSGI maintains a list of those jobs willing to accept input. MSGI stores characters input from the console in one of several input buffers in zone 4. Upon completion of the input

(indicated by receipt of an ETX or carriage return) the job receiving the input is called. After the job call, register R3 will contain the starting address of the input text and register R2 will contain the value 0045_{16}. The end of the input text is delimited by an ETX.

The job receiving input has exclusive use of the input buffer until it has completed analyzing the input text. When the job no longer needs the input text, the buffer must be released.

No more than one input request may be outstanding from any one job. Additional requests are discarded. An input request is required for each console input.

To receive input from the console:

XOP @MSGI,TRMIO
RETURN - INPUT REQUEST ACCEPTED
AFTER THE JOB CALL OF SUSPENDED JOB
R3 = STARTING ADDRESS OF INPUT TEXT
R2 = 0045_{16}

4.4.3 Radio/1822 DMA I/O

Twelve DMA I/O functions are provided via XOP call as shown below:

- Initiate Radio RX Channel 1 or 2 DMA I/O (RDRX)
- Initiate Radio RX Channel 1 DMA I/O (RDRX1)
- Initiate Radio RX Channel 2 DMA I/O (RDRX2)
- Initiate Radio TX DMA I/O (RDTX)
- Initiate 1822 RX DMA I/O (STRX)
- Initiate 1822 TX DMA I/O (STTX)
- Idle Radio RX Channel 1 or 2 DMA I/O (IRDRX)
- Idle Radio RX Channel 1 DMA I/O (IRDRX1)
- Idle Radio RX Channel 2 DMA I/O (IRDRX2)
- Idle Radio TX DMA I/O (IRDTX)
- Idle 1822 RX DMA I/O (ISTRX)
- Idle 1822 TX DMA I/O (ISTTX)

The XOP vector name and event codes are listed below:

```
DMAIO    EQU     5
RDRX     DATA    >0052
RDRX1    DATA    >0053
RDRX2    DATA    >0054
RDTX     DATA    >0055
STRX     DATA    >0050
STTX     DATA    >0051
IRDRX    DATA    >005A
IRDRX1   DATA    >005B
IRDRX2   DATA    >005C
IRDTX    DATA    >005D
ISTRX    DATA    >0058
ISTTX    DATA    >0059
```

4.4.3.1 Initiate Radio RX DMA I/O

To initiate a Radio RX, the user must first secure a packet buffer for the radio transaction and then insert the Radio RX DMA control parameter in the IPR DMA I/O Packet Buffer Overhead. The Radio RX control parameter is illustrated in Figure 12.

After the packet buffer is obtained and the control word is stored, the user can also select the automatic re-initiate Radio RX DMA on error option by loading the Retry Limit byte of the Retry On Error register in the DMA I/O Packet Buffer Overhead with a value greater than zero. The ENABLE (bit 15) of the control word must be zero. At this point, the following call is used:

```
R3 = PACKET BUFFER ADDRESS WHICH WILL CONTAIN THE DATA
     FROM RADIO RECEPTION
XOP @RDRX,DMAIO  FOR RADIO RX CHANNEL 1 OR 2
XOP @RDRX1,DMAIO FOR RADIO RX CHANNEL 1 ONLY
XOP @RDRX2,DMAIO FOR RADIO RX CHANNEL 2 ONLY
RETURN - RADIO RX DMA I/O REQUEST QUEUED AND/OR INITIATED
```

The Radio RX DMA Word Count hardware register is loaded with the maximum packet length of 129 words, which includes header + text + 2 CRC words. The Radio RX DMA Address hardware register is loaded with the Radio TX DMA I/O packet buffer address passed in R3. The Radio RX DMA Control register, which is located in the DMA I/O Packet Buffer Overhead, is passed to the Radio RX DMA Control/Status hardware register with bit 15 (ENABLE) set. At this point the Radio RX DMA I/O request is initiated.

BITS	FUNCTION	DEFINITION
0-12	Not Used	--
13	TST CHNL SEL	0 - Normal Operation 1 - Loop Test of Receiver Interface and Channel Selector
14	TST RX I/F	0 - Normal Operation 1 - Loop test of RX interface
15	ENABLE	0 - DMA disabled 1 - DMA enabled

FIGURE 12 RADIO RX CONTROL WORD

Once the Radio RX DMA hardware interface generates an interrupt, the contents of the final Word Count, Address and Control/Status hardware registers are saved in the IPR DMA I/O Packet Buffer Overhead. Since the value of the Radio RX DMA Word Count hardware register is decremented during the reception of data from the Radio Unit of the PR to the Digital Unit, a value of 129 minus the length of the received packet at the time of the interrupt would indicate a successfully complete Radio RX DMA reception. The Radio RX DMA Address hardware register will contain the value of the last address +2 written at the time of completion. The Radio RX DMA Status word is described in Figure 13.

If the Radio RX DMA Control/Status hardware register indicates that the DMA transaction had error(s), or if the DMA transaction did not fully complete (length of received packet is not equal to packet length); the Error Count byte of the DMA Retry on Error register in the DMA I/O Packet Buffer Overhead is incremented. If this Error Count is less than or equal to the Retry Limit byte of the Retry on Error register, the Radio RX DMA I/O request is re-initiated. If there are no error(s) or the Error Count is greater than the Retry Limit a JOB CALL RETURN is then issued to the job which initiated the Radio RX DMA I/O request. The job's workspace registers will be loaded with the following information at the time of the call:

```
R2 = RADIO RX DMA I/O EVENT CODE
     (SAME AS USED IN INITIATE CALL)
R3 = PACKET BUFFER ADDRESS WHICH CONTAINS
     THE RECEIVED PACKET
```

4.4.3.2 Initiate Radio TX DMA I/O

To initiate a Radio TX, the user must first secure a packet buffer for the radio transaction and then insert the Radio TX DMA control parameter in the IPR DMA I/O Packet Buffer Overhead. The Radio TX control parameter is illustrated in Figure 14.

After the packet buffer is obtained and the control word is stored, the user can also select the automatic re-initiate Radio TX DMA on error option by loading the Retry Limit byte of the Retry on Error register in the DMA I/O Packet Buffer Overhead with a value greater than zero. The ENABLE (bit 15) of the control word must be zero. At this point, the following call is used:

```
R3 = PACKET BUFFER ADDRESS WHICH CONTAINS THE DATA
     FOR THE RADIO TRANSMISSION
 XOP  @RDTX,DMAIO
 RETURN - RADIO TX DMA I/O REQUEST QUEUED AND/OR INITIATED
```

BITS	FUNCTION	DEFINITION
0-4	Not used	
5	CHANNEL SEL	0 => Channel A 1 => Channel B
6	RX TERM	Radio RX terminated due to radio TX
7	EOP SYNC	Received EOP
8	HIGH DATA RATE	0 => 100 Kbit/sec 1 => 400 Kbit/sec
9	OVERRUN ERROR	Memory overrun error
10	SYNC ERROR	Loss of EOP error
11	CRC ERROR	Checksum error
12	MEMORY ERROR	Memory parity error and/or DMA bus time-out error
13	WD COUNT ERROR	Incorrect word count loaded into DMA word count register
14	DONE	DMA operation completed
15	ENABLE	Image of radio RX DMA ENABLE control bit

FIGURE 13 RADIO RX STATUS WORD

BITS	FUNCTION	DEFINITION
0-5	FREQ CTRL	Image of RF frequency bits defined in Section 2.11.1.4
6	TX RANDOMIZE	0 => Non Randomize Radio TX 1 => Randomize radio TX
7	DISC ALOHA	0 => Aloha 1 => Disciplined Aloha
8	HIGH DATA RATE	0 => 100 Kbit/sec 1 => 400 Kbit/sec
9-11	RF POWER	111 => Full power 110 => -5 dB 101 => -10 dB 011 => -15 dB 001 => -20 dB
12	ENABLE RX	0 => Disable radio RX 1 => Enable radio RX
13	TX CS MODE	0 => Aloha modes 1 => Carrier sense modes
14	TEST	0 => Normal operation 1 => Loop test
15	ENABLE	0 => DMA disabled 1 => DMA enabled

FIGURE 14 RADIO TX CONTROL WORD

The Radio TX DMA Control register, which is located in the DMA I/O Packet Buffer Overhead, is passed to the Radio TX DMA Control/Status hardware register 140 microseconds before bit 15 (ENABLE) is set. This allows the RF hardware to set the frequency, data rate and power level before the DMA is initiated. The contents of the Radio TX Randomization Time register in the DMA I/O Packet Buffer Overhead is loaded into the Radio TX Randomize hardware register. The Radio TX DMA Word Count hardware is loaded with the value obtained in the least significant byte of the first word of the header + 3 words to include the preamble. The Radio TX DMA Address hardware register is loaded with the Radio RX DMA I/O packet buffer address - 3 to include the transmission of the preamble along with the header and text. At this point the Radio TX DMA Control register is passed to the hardware with the ENABLE bit set -- the Radio TX DMA I/O request is now initiated.

Once the Radio TX DMA hardware interface generates an interrupt, the contents of the final Word Count, Address and Control/Status hardware registers are saved in the IPR DMA I/O Packet Buffer Overhead. Since the value of the Radio TX DMA Word Count hardware register is decremented during the transmission of data from the Digital Unit of the PR to the Radio Unit, a value of zero at the time of the interrupt would indicate a successfully complete Radio TX DMA transmission. The Radio TX DMA Address hardware register will contain the value of the last address + 2 read at the time of completion. The Radio TX DMA Status word is described in Figure 15.

If the Radio TX DMA Control/Status hardware register indicates that the DMA transaction had error(s), or if the DMA transaction did not fully complete (length of transmitted packet is not equal to packet length); the Error Count byte of the DMA Retry on Error register in the DMA I/O Packet Buffer Overhead is incremented. If this error count is less than or equal to the Retry Limit byte of the Retry on Error register, the Radio TX DMA I/O request is re-initiated. If there are no error(s) or the Error Count is greater than the Retry Limit a JOB CALL RETURN is then issued to the job which initiated the Radio TX DMA I/O request. The Job's workspace registers will be loaded with the following information at the time of the call:

R2 = ADIO TX DMA I/O EVENT CODE
(SAME AS USED IN INITIATE CALL)
R3 = PACKET BUFFER ADDRESS WHICH CONTAINS
THE TRANSMITTED PACKET

BITS	FUNCTION	DEFINITION
0-10	Not used	
11	OVERRUN ERROR	Memory overrun error
12	MEMORY ERROR	Memory parity and/or DMA bus time-out error
13	Not used	
14	DONE	DMA operation completed
15	ENABLE	Image of radio TX DMA enable control bit

FIGURE 15 RADIO TX STATUS WORD

4.4.3.3 Initiate 1822 RX DMA I/O

To initiate an 1822 RX, the user must first secure a packet buffer for the station transaction and then insert the 1822 RX DMA control parameter in the IPR DMA I/O Packet Buffer Overhead. The 1822 RX control parameter is illustrated in Figure 16.

After the packet buffer is obtained and the control word is stored, the user can also select the automatic re-initiate 1822 RX DMA on error option by loading the Retry Limit byte of the Retry on Error register in the DMA I/O Packet Buffer Overhead with a value greater than zero. The ENABLE (bit 15) of the control word must be zero. At this point, the following call is used:

```
R3 = PACKET BUFFER ADDRESS WHICH WILL CONTAIN THE DATA
     FROM THE STATION RECEPTION
XOP  @STRX,DMAIO
RETURN - 1822 RX DMA I/O REQUEST QUEUED AND/OR INITIATED
```

BITS	FUNCTION	DEFINITION
0-13	Not used	
14	TEST	0 => Normal operation 1 => Loop test
15	ENABLE	0 => DMA disabled 1 => DMA enabled

FIGURE 16 1822 RX CONTROL WORD

The 1822 RX DMA Word Count hardware register is loaded with the maximum packet length of 129 words, which includes header + text + 2 CRC words. The 1822 RX DMA Address hardware register is loaded with the 1822 TX DMA I/O packet buffer address passed in R3. The 1822 RX DMA Control register, which is located in the DMA I/O Packet Buffer Overhead, is passed to the 1822 RX DMA Control/Status hardware register with bit 15 (ENABLE) set. At this point the 1822 RX DMA I/O request is initiated.

Once the 1822 RX DMA hardware interface generates an interrupt, the contents of the final Word Count, Address and Control/Status hardware registers are saved in the IPR DMA I/O Packet Buffer Overhead. Since the value of the 1822 RX DMA Word Count hardware register is decremented during the reception of data from the station to the Digital Unit of the PR, a value of 129 minus the length of the received packet at the time of the interrupt would indicate a successfully complete 1822 RX DMA reception. The 1822 RX DMA Address hardware register will contain the value of the last address + 2 written at the time of completion. The 1822 RX DMA Status word is described in Figure 17.

If the 1822 RX DMA Control/Status hardware register indicates that the DMA operation had error(s), or if the DMA transaction did not fully complete (length of received packet is not equal to packet length), the Error Control byte of the DMA Retry on Error register in the DMA I/O packet buffer overhead is incremented. If the Host Ready bit is set (HOST READY), then the flap routine is initiated (PR RDY bit of the 1822 TX DMA control word is cleared and set, with a delay of one second between these operations) and processing continues. If this error count is less than or equal to the retry limit byte of the retry on error register, the 1822 RX DMA I/O request is re-initialized. If there are no error(s) or the error count is greater than the retry limit a JOB CALL RETURN is immediately issued to the job which initiated the 1822 RX DMA I/O request. The job's workspace registers will be loaded with the following information at the time of the call:

R2 = 1822 RX DMA I/O EVENT CODE
(SAME AS USED IN INITIATE CALL)
R3 = PACKET BUFFER ADDRESS WHICH CONTAINS
THE RECEIVED PACKET

BITS	FUNCTION	DEFINITION
0-5	Not used	
6	LAST BIT ERROR	Last bit did not occur on a 16-bit word boundary
7	Not used	
8	HOST READY	Host ready
9	Not used	
10	READY DROPPED	Host ready dropped
11	CRC ERROR	Checksum error
12	MEMORY ERROR	Memory parity and/or DMA bus time-out error
13	WORD COUNT ERR	Incorrect word count loaded into DMA word count register
14	DONE	DMA operation completed
15	ENABLE	Image of 1822 RX DMA enable control bit

FIGURE 17 1822 RX STATUS WORD

4.4.3.4 Initiate 1822 TX DMA I/O

Before initiating an 1822 TX, the user should verify the ready condition of the 1822 interface. The PR RDY/STA RDY can be checked using the STIO XOP as described in Section 4.4.3.10. 1822 TX initiation should be delayed until the interface is ready.

To initiate an 1822 TX, the user must first secure a packet buffer for the station transaction and then insert the 1822 TX DMA control parameter in the IPR DMA I/O Packet Buffer overhead. The 1822 TX control parameter is illustrated in Figure 18.

After the packet buffer is obtained and the control word is stored, the user can also select the automatic re-initiate 1822 TX DMA on error option by loading the Retry Limit byte of the Retry on Error register in the DMA I/O Packet Buffer Overhead with a value greater than zero. The ENABLE (bit 15) of the control word must be zero. At this point, the following call is used:

```
R3 = PACKET BUFFER ADDRESS WHICH CONTAINS THE DATA
     FOR THE STATION TRANSMISSION
XOP @STTX,DMAIO
RETURN - 1822 TX DMA I/O REQUEST QUEUED AND/OR INITIATED
```

The 1822 TX DMA Word Count hardware register is loaded with the value obtained in the least significant byte of the first word of the header + 3 words to include the preamble. The 1822 TX DMA Address hardware register is loaded with the 1822 TX DMA I/O packet buffer address - 3 to include the transmission of the preamble along with the header and text. The 1822 TX Control register, which is located in the DMA I/O Packet Buffer Overhead, is examined. If the TEST bit is not set, indicating normal 1822 operation, the PR RDY bit is set. At this point, the 1822 TX DMA Control register is passed to the hardware with the ENABLE bit set -- the 1822 TX DMA I/O request is now initiated.

Once the 1822 TX DMA hardware interface generates an interrupt, the contents of the final Word Count, Address and Control/Status hardware registers are saved in the IPR DMA I/O Packet Buffer Overhead. Since the value of the 1822 TX DMA Word Count hardware register is decremented during the transmission of data from the Digital Unit of the PR to the station, a value of zero at the time of the interrupt would indicate a successfully complete 1822 TX DMA transmission. The 1822 TX DMA Address hardware register will contain the value of the last address + 2 read at the time of completion. The 1822 TX DMA Status word is described in Figure 19.

BITS	FUNCTION	DEFINITION
0-12	Not used	
13	PR READY	0 => PR not ready 1 => PR ready
14	TEST	0 => Normal operation 1 => Loop test
15	ENABLE	0 => DMA disabled 1 => DMA enabled

FIGURE 18 1822 TX CONTROL WORD

BITS	FUNCTION	DEFINITION
0-7	Not used	
8	HOST READY	Host ready
9	IMP READY	Ready for IMP bit
10	Not used	
11	TX RDY ERROR	While transmitting a pkt, the host ready dropped
12	MEMORY ERROR	Memory parity and/or DMA bus time-out error
13	Not used	
14	DONE	DMA operation completed
15	ENABLE	Image of 1822 TX DMA enable control bit

FIGURE 19 1822 TX STATUS WORD

If the 1822 TX DMA Control/Status hardware register indicates that the DMA transaction had error(s), or if the DMA transaction did not fully complete (length of transmitted packet is not equal to packet length), the Error Count byte of the DMA Retry On Error register in the DMA I/O Packet Buffer Overhead is incremented. If the Host Ready bit is set (HOST READY), then the flap routine is initiated (PR RDY bit of the 1822 TX DMA Control word is cleared and set, with a delay of one second between these operations) and processing continues. If this Error Count is less than or equal to the Retry Limit byte of the Retry on Error register, the 1822 TX DMA I/O request is re-initiated. If there are no error(s) or the Error Count is greater than the Retry Limit a JOB CALL RETURN is immediately issued to the job which initiated the 1822 TX DMA I/O request. The job's workspace registers will be loaded with the following information at the time of the call:

```
R2 = 1822 TX DMA I/O EVENT CODE
     (SAME AS USED IN INITIATE CALL)
R3 = PACKET BUFFER ADDRESS WHICH CONTAINS
     THE TRANSMITTED PACKET
```

4.4.3.5 Idle Radio RX DMA I/O

To idle a previously initiated Radio RX DMA, the following call is used:

```
R3 = PACKET BUFFER ADDRESS OF RADIO RX DMA TO BE IDLED
XOP @IRDRX,DMAIO FOR RADIO RX CHANNEL 1 OR 2
XOP @IRDRX1,DMAIO FOR RADIO RX CHANNEL 1 ONLY
XOP @IRDRX2,DMAIO FOR RADIO RX CHANNEL 2 ONLY
RETURN - RADIO RX DMA I/O REQUEST IDLED
```

4.4.3.6 Idle Radio TX DMA I/O

To idle a previously initiated Radio TX DMA, the following call is used:

```
R3 = PACKET BUFFER ADDRESS OF RADIO TX DMA TO BE IDLED
XOP @IRDTX,DMAIO FOR RADIO TX
RETURN - RADIO TX DMA I/O REQUEST IDLED
```

4.4.3.7 Idle 1822 RX DMA I/O

To idle a previously initiated 1822 RX DMA, the following call is used:

```
R3 = PACKET BUFFER ADDRESS OF 1822 RX DMA TO BE IDLED
XOP @ISTRX,DMAIO FOR 1822 RX
RETURN - 1822 RX DMA I/O REQUEST IDLED
```

4.4.3.8 Idle 1822 TX DMA I/O

To idle a previously initiated 1822 TX DMA, the following call is used:

```
R3 = PACKET BUFFER ADDRESS OF 1822 TX DMA TO BE IDLED
XOP @ISTTX,DMAIO FOR 1822 TX
RETURN - 1822 TX DMA I/O REQUEST IDLED
```

4.4.3.9 1822 I/O Check Station Ready

The 1822 I/O Check Station Ready (STIOR) routine provides the following services:

- Set PR RDY bit of 1822 hardware Control register
- Check HOST RDY bit of 1822 hardware Status register and check STA RDY flag

The XOP vector name and event code for 1822 I/O Check Station Ready are as follows:

```
STIO    EQU     8
STIOR   DATA    >0080
```

To initiate an 1822 I/O Check Station Ready, the following call is issued by the user:

```
XOP @STIOR,STIO
RETURN - STATION NOT READY
RETURN + 4 - STATION READY
```

Once an 1822 I/O Check is issued, the STA RDY flag is checked. If it is not ready, a process is initiated which will set the PR RDY bit (13) of the 1822 TX Control register after one second and will set the STA RDY flag to ready after another second. After the process is initiated, the 1822 I/O Check routine is exited without adjusting the program counter (PC). If the STA RDY flag is ready, then the HOST RDY bit (8) in the 1822 TX Status register is checked.

If HOST RDY is set (Host is ready), then the program counter is adjusted (PC+4) and the 1822 I/O check routine is exited.

If HOST RDY is zero (Host not ready), then the routine is exited without adjusting the program counter (PC).

4.4.3.10 1822 I/O Clear Station Ready

The 1822 I/O Clear Station Ready (STIOCL) routine clears the PR RDY bit of the 1822 hardware Control register.

The XOP vector name and event code for 1822 I/O Clear Station Ready are as follows:

```
STIO      EQU     8
STIOCL    DATA    >0081
```

To initiate an 1822 I/O Clear Station Ready, the following call is issued by the user:

```
XOP @STIOCL,STIO
RETURN
```

4.4.3.11 IPR DMA Packet Buffer Overhead

The IPR DMA Packet Buffer Overhead format is illustrated in Figure 20:

RESERVED

One word has been reserved for future expansion of the IPR Packet Buffer Overhead.

DMA I/O FORWARD POINTER

Pointer to next DMA I/O event packet buffer in the DMA linked-list (queue).

DMA I/O CALLING PROCESS I/D

Job I/D and Event Code of current DMA I/O event. (Read by User).

DMA WORD COUNT REGISTER

Image of DMA Word Count hardware register when DMA interrupt or idle occurs. (Read by User)

DMA ADDRESS REGISTER

Image of DMA Address hardware register when DMA interrupt or idle occurs. (Read by User)

DMA STATUS REGISTER

Image of DMA Status hardware register when DMA interrupt or idle occurs. (Read by User)

Offset	Field
-48	RESERVED
-46	DMA I/O FORWARD POINTER
-44	DMA I/O CALLING PROCESS I/D
-42	DMA WORD COUNT REGISTER
-40	DMA ADDRESS REGISTER
-38	DMA STATUS REGISTER
-36	DMA CONTROL PARAMETERS
-34	RADIO TX RANDOMIZATION TIME
-32	DMA I/O RETRY ON ERROR
-30	DMA I/O EVENT CLOCK (00-15)
-28	DMA I/O EVENT CLOCK (16-31)
-26	DMA I/O STATUS
-12 / -24	USER SPACE 1 / 7
-10	PACKET BUFFER ASSIGN/RELEASE
- 8	PACKET BUFFER CHAIN ADDRESS
- 2 / - 6	PACKET PREAMBLE 1 / 3
0	PACKET HEADER / TEXT : : :

FIGURE 20 IPR DMA PACKET BUFFER OVERHEAD

DMA CONTROL PARAMETER

Contains DMA Control word created user and/or DMA I/O initiate routine, which will be loaded into the DMA Control/Status hardware register. Used to initiate and stop/clear radio and 1822 DMA hardware. (Provided by User)

RADIO TX RANDOMIZATION TIME

Contains randomization time for Radio TX DMA I/O initiate events. This value of time is loaded into the Radio TX Randomize hardware register before the Radio TX DMA hardware is enabled for transmission. (Provided by User).

DMA I/O RETRY ON ERROR

Contains retry limit set by user and error count by DMA channel. (Retry Limit is Provided by User -- Maximum of 255).

DMA I/O EVENT CLOCK

Loaded with contents of the 32-bit Elapsed Timer each time I/O Status is updated. (Timer is binary count incremented at 6.51 microsecond rate).

DMA I/O STATUS

Indicates current processing status of the packet buffer. Figure 21 defines the IPR I/O Status. (Read by User).

USER SPACE

Seven words reserved to be defined and used by the high level programs which used the packet buffers for packet I/O processing. (For User's Own Needs).

PACKET BUFFER ASSIGN/RELEASE

Indicates if packet buffer is presently assigned to a job or resides on the packet buffer queue. (Read by User).

PACKET BUFFER CHAIN ADDRESS

Points to next packet buffer address in the packet buffer queue.

PACKET PREAMBLE

Contains the 3 words of preamble used by the packet radio network.

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
                                  |   |   |   |   |   |   |   |
                                  |   |   |   |   |   |   |   QUEUED
                                  |   |   |   |   |   |   |
                                  |   |   |   |   |   |   INITIATED
                                  |   |   |   |   |   |
                                  |   |   |   |   |   ENABLED
                                  |   |   |   |   |
                                  |   |   |   |   INTERRUPTED
                                  |   |   |   |
                                  |   |   |   SERVICE COMPLETED
                                  |   |   |
                                  |   |   SERV COMP'D WITH ERROR(S)
                                  |   |
                                  |   RADIO RX HOLD / 1822 ERROR
                                  |
                                  IDLED
```

FIGURE 21 IPR DMA I/O STATUS

PACKET HEADER/TEXT

Contains packet body. First word of the header contains the packet length (bits 8 through 15, least significant byte) in words. Packet address passed in Radio/1822 DMA I/O call (defines first word of packet header text). (Defined by User).

4.4.4 Buffer Allocation

This call is used by the system jobs to obtain dynamic buffer space for job use and to release the buffer space for reassignment when it is no longer needed by the job. The allocated buffer space is in zone 3 memory.

At present, two calls are implemented. These are:

Assign packet buffer (ASGPB)
Release packet buffer (RLSPB)

The XOP call arguments are shown below:

```
BUFFER  EQU    3
ASGPB   DATA   >0030
RLSPB   DATA   >0031
```

4.4.4.1 Assign Packet Buffer

This call is used to request a packet buffer. Two immediate returns are provided: the first to indicate no buffers are available for assignment, the second to indicate assignment of a buffer. The buffer address is returned to the caller's workspace register R3 and defines the first word of the packet header/text. The call is shown below:

```
XOP   @ASGPB,BUFFER
RETURN    - NO BUFFER AVAILABLE
RETURN+4  - ASSIGNED BUFFER ADDRESS IN R3
```

4.4.4.2 Release Packet Buffer

This call is used to release a previously assigned packet buffer. The address of the packet buffer to be release is contained in the calling job's workspace register R3. The operating system will halt the calling job if the packet buffer address is erroneous, if the DMA I/O activity for which the packet is being used has not completed or been idled, or if the packet buffer has already been released. The call is shown below:

```
R3 = PACKET BUFFER ADDRESS
XOP   @RLSPB,BUFFER
RETURN - PACKET BUFFER RELEASED
```

4.4.5 Interjob Communication

This call provides the mechanism for interjob communication. It is used for initial job calls and job call returns in response to an initial job call.

An interjob call is defined by a job call ID and a data word. The job call ID consists of a job ID (most significant byte) and an event code (least significant byte) defined by the called job. The job call ID is defined by the XOP instruction. The optional data word, typically a buffer address, is defined in the calling job's workspace register R3. Note that only zone 3 buffer is accessible by all jobs. Local zone 2 buffer of a job is not directly accessible by other jobs due to bias register 2 modification by the operating system. The interjob call is illustrated below. JBCALL (01) is the XOP vector name.

```
R3 = OPTIONAL DATA WORD
XOP  @(job call ID),JBCALL
RETURN   - CALLED JOB NOT RESIDENT
RETURN+4 - JOB CALL ISSUED
```

The operating system replaces the called job's ID with the calling job's ID and inverts the most significant bit of the event code in the job call ID. This, with the data word, is placed in the called job's workspace register R2 and R3 when it is initiated from the point of suspension. The content of R2 is the job call ID to be used for recall, if necessary, of the original job when the event processing is completed. The job call return will return the original job call ID and data word to the initial calling job.

The called job decodes the event code. The most significant bit of the event code, if one, is a request to perform event processing by the job. If zero, it defines a completed event previously requested by the job.

4.4.6 Utility Service

Four utility functions are provided via XOP call as shown below:

- Update enabled CPU's execution vector (CPUV)
- Get CPU's configuration table address (CCADDR)
- Move error data to buffer (MOVER)
- General Purpose AMDM (GPAMDM)

The XOP vector name and event codes are listed below:

```
UTIL     EQU     7
CPUV     DATA    >0070
CCADDR   DATA    >0071
MOVER    DATA    >0072
GPAMDM   DATA    >0073
```

These functions are used by the operating system and off-line diagnostic jobs.

4.4.6.1 Update Enabled CPU Execution Vector

This call is utilized to enable and disable operation of implemented CPUs. A CPU when disabled will idle at interrupt level 3. It is used by diagnostic jobs to restrict execution (testing) to a single CPU. The desired vector is passed in the least significant byte of workspace register R3. Two immediate returns are provided: the first to indicate that the System Job List entry was not updated because all the CPUs represented in the argument were not implemented; the second to indicate that the entry was updated. The call is shown below:

```
R3 = DESIRED CPU EXECUTION VECTOR
XOP  @CPUV,UTIL
RETURN   - SYSTEM JOB LIST ENTRY NOT UPDATED
RETURN+4 - SYSTEM JOB LIST ENTRY UPDATED
```

R3 bit 15 defines CPU 1. R3 bit 14 defines CPU 2. Each bit if 1 enables, or if 0 disables, execution by the corresponding CPU.

4.4.6.2 Get CPU Configuration Table Address

This call provides the absolute zone 4 address of the executing CPU's Configuration Table address thereby providing access to the data contained therein. The call is shown below:

```
XOP  @CCADDR,UTIL
RETURN - ADDRESS IN REGISTER R3
```

4.4.6.3 Move Error Data to Buffer

This call defines summary error data generated by a job or the operating system. The process moves the error data to an operating system defined buffer. This data is subsequently inserted in Load Request ROP packets to report PR failure to a remote facility.

The number of error data words (equal to or less than 21_{10}) is defined in register R2. The start address of the data is defined in R3. Data in ascending addresses is moved to the buffer. The call is illustrated below:

```
R2 = NUMBER OF 16-BIT WORDS
R3 = START ADDRESS OF WORDS
XOP  @MOVER,UTIL
RETURN - DATA MOVED TO ERROR BUFFER
```

4.4.6.4 General Purpose AMDM

This call will change a memory location or transfer the contents of a memory location to the user's buffer. Operation of the call is defined by the contents of R4, which will contain Ø for alter memory (AM) or 2 for display memory (DM).

The address of the buffer to be used is defined in R2. The HI and LO address of the AMDM location is defined in R5 and R6. The AMDM address can be of three types; CPU, BUS, or Relocatable job. The type of address is defined by the contents of R5 as illustrated below:

R5 CONTENTS

ADDR TYPE	HI BYTE	LO BYTE
CPU	Ø	80 HEX
BUS	Ø	HI 4 bits of ADDR
Reloc. Job	Job ID	80 HEX

The remainder of the address is defined by the contents of R6 for either of the three cases. The call is illustrated below:

```
R2 = ADDRESS OF BUFFER LOCATION
R4 = OPERATION FLAG - Ø ALTER MEMORY
                      2 DISPLAY MEMORY
R5 = HI ADDR OF AMDM LOCATION
R6 = LO ADDR OF AMDM LOCATION
XOP @GPAMDM,UTIL
RETURN - OPERATION COMPLETED
         BUFFER ADDRESS INCREMENTED
         AMDM LOCATION ADDRESS INCREMENTED
```

4.4.7 I/O TAG READ/WRITE SERVICE

Seven I/O TAG functions are provided via XOP call as shown below:

- Read MS word of elapsed timer (RDMS)
- Read LS word of elapsed timer (RDLS)
- Read both words of elapsed timer (RDBO)
- Reset elapsed timer (RESETT)
- Read PR ID straps (RDID)
- Read RAM image of MODE straps (RDMODE)
- Write to front panel CODE and DATA LEDS (WRPNL)

The XOP vector name and event codes are listed below:

```
IOTAG     EQU     10
RDMS      DATA    >XXA0
RDLS      DATA    >XXA1
RDBO      DATA    >XXA2
RESETT    DATA    >XXA3
RDID      DATA    >XXA4
RDMODE    DATA    >XXA5
WRPNL     DATA    >XXA6
```

Where XX equals the word offset from the beginning of the caller's workspace.

4.4.7.1 <u>Read MS Word of Elapsed Timer</u>

This call will return the MS word of the elapsed timer in the user's workspace register indicated by the MS byte of the event code. The call is shown below:

```
XX = USER'S WORKSPACE REGISTER
EC  DATA >XXA0
XOP  @EC,IOTAG
RETURN - ELAPSED TIME IN WORKSPACE REGISTER XX
```

4.4.7.2 Read LS Word of Elapsed Timer

```
XX = USER'S WORKSPACE REGISTER
EC  DATA >XXA1
XOP  @EC,IOTAG
RETURN - ELAPSED TIME IN WORKSPACE REGISTER XX
```

4.4.7.3 <u>Read Both Words of Elapsed Timer</u>

This call will return the MS word of the elapsed timer in the user's workspace register indicated by the MS byte of the event code. The LS word of the elapsed timer will be returned in the next higher workspace register. The MS word returned is checked to be sure it is consistent with a LS word that may have just rolled over. The call is shown below:

```
XX = USER'S WORKSPACE REGISTER
EC  DATA  >XXA2
XOP @EC,IOTAG
RETURN - MS WORD OF ELAPSED TIME IN WORKSPACE REGISTER XX
       - LS WORD OF ELAPSED TIME IN WORKSPACE REGISTER XX+1
```

4.4.7.4 Reset Elapsed Timer

This call will reset the elapsed timer to zero. The call is shown below:

```
XX = DON'T CARE
EC  DATA >XXA3
XOP @EC,IOTAG
RETURN - ELAPSED TIMER RESET
```

4.4.7.5 Read PR ID Straps

This call will return the contents of the PR ID straps in the user's workspace register indicated by the MS byte of the event code. The call is shown below:

```
XX = USER'S WORKSPACE REGISTER
EC  DATA >XXA4
XOP @EC,IOTAG
RETURN - PR ID CODE IN WORKSPACE REGISTER XX
```

4.4.7.6 Read RAM Image of MODE Straps

This call will return the contents of the RAM image of the MODE straps in the user's workspace register indicated by the MS byte of the event code. The call is shown below:

```
XX = USER'S WORKSPACE REGISTER
EC  DATA >XXA5
XOP @EC,IOTAG
RETURN - MODE CODE IN WORKSPACE REGISTER XX
```

4.4.7.7 Write to Front Panel CODE and DATA LEDS

This call will write the contents of two consecutive user workspace registers to the CODE and DATA LEDS respectively of the front panel. The user's workspace registers are indicated by the MS byte of the event code. The call is shown below:

```
XX = USER'S WORKSPACE REGISTER
EC  DATA >XXA6
XOP @EC,IOTAG
RETURN - CONTENTS OF WORKSPACE REGISTER XX WRITTEN TO CODE LEDS
       - CONTENTS OF WORKSPACE REGISTER XX+1 WRITTEN TO DATA
         LEDS
```

4.4.8 Down Load Service

Two functions are provided to facilitate down loading of the OS code or protocol jobs. They are provided by XOP call as shown below:

- Fetch a word of OS or Protocol data (DLLGET)
- Insert name of OS into Packet Buffer (DLLOSI)

The XOP vector name and event codes are listed below:

```
DLLSUB     EQU      9
DLLGET     DATA     >90
DLLOSI     DATA     >91
```

4.4.8.1 Fetch a Word of OS or Protocol Data

This call will fetch a word of OS or Protocol data based upon the protocol/OS flag and a sequence #. The call is shown below:

```
R5 = WORD #
R6 = PROTOCOL/OS FLAG
XOP  @DLLGET,DLLSUB
Return - R5 & R6 UNCHANGED
       - R7 = FETCHED WORD
       - R8 = TAGFLAG (DATA/ADDR)
       - R9 = BEGJOB (START OF JOB FLAG)
       - R0 = ENDJOB (END OF JOB)
       - R1 = ENDLOD (END OF LOAD)
```

4.4.8.2 Insert Name of OS into Packet Buffer

This call will insert the name of the loaded part of the OS into the users packet buffer. The call is shown below:

```
R4 = PACKET BUFFER INPUT POINTER
XOP  @DLLOSI,DLLSUB
RETURN - OS NAME IN PACKET BUFFER
```

4.4.9 Operator Control/Monitor (Debug) Job Call

A job may issue a call to the operating system debug job. The call defines an operator command or set of chained commands as defined in Section 5.4. The call is shown below:

```
R3 = ADDRESS OF ASCII CHARACTER STRING
XOP  @job call ID,JBCALL
RETURN   - DEBUG NOT RESIDENT (System Failure)
RETURN+4 - JOB CALL ISSUED
```

JBCALL the vector name equals 1. The job call ID word equals 0500_{16}. The job's workspace register R3 defines the start address of the command ASCII character string. The ASCII character string is stored in the order left byte/right byte in increasing memory addresses terminated by ETX in the zone 3 address space.

5 OPERATING PROCEDURES

The following paragraphs describe operator actions and displays visible to the operator during operation of the PR.

5.1 Operation/Hardware Strap Options

The hardware straps which effect PR operation are described in Section 2.11. The terminal type and terminal baud rate straps must be set for the console connected to the PR I/O channel. The PR ID straps are set to uniquely define the PR. The auto-restart and down-line load straps are set to define the appropriate operational configuration as described below. The radio RF frequency straps are set to define the value used for down-line loading of the PR. User jobs may use this value or other values. The initialization mode straps are normally zero except when initialization faults need to be debugged.

For changed values of the auto-restart, down-line load, and/or radio RF frequency switches to take effect, power-up initialization or a restart must be (re)executed, i.e., the INIT or RESET pushbutton must be pushed or the IN or RS console command must be executed.

5.1.1 Normal Unattended Operation

For normal PR operation, unattended by an operator, auto-restart and down-line load are enabled via the hardware straps. The PR I/O channel console need not be implemented except for use as a monitor display. After initial initialization invoked by the operator, down-line loading of the remaining portion of the operating system and the user jobs and initiation of job execution is done automatically. The PR will auto-restart and if necessary reload software in the event of failure.

Operational sequence

Operator action	Depress and release front panel initiate (INIT) pushbutton.
Display	Console bell, then LED display of sequential execution of initialization sections 1-6 (01-06 in code LEDs) by each CPU (N000 in data LEDs, where N is the CPU number). Blinking display if errors. See Section 5.2.

Display LED display of initialization section 7 by one CPU plus console message for down-line load of operating system:

DOWN-LINE LOAD-END

Blinking LEDs and console messages if error. See Section 5.2 and Section 5.4.17.

Display LED display of initialization section 8 by each CPU (08 in code LEDs, N000 in data LEDs). Blinking display if errors. See Section 5.2.

Display LED display (09 N000) and console messages for initialization and job loading:

DOWN-LINE LOAD-END

Blinking LEDs and console messages if error. See Section 5.2 and Section 5.4.17.

Display Final LED display for successful initialization is 0F 0000. Console message for Debug job initiation:

JOB DEBUG

Display: Console message for initiation of job execution:

AUTO-XJ

LED display is blanked (turned off).

5.1.2 Normal Attended Operation

For normal PR operation, attended by an operator, auto-restart is enabled and down-line load is disabled via hardware straps. The remaining portion of the operating system and the user jobs are loaded by the operator via the PR I/O channel console. Job execution is initiated via the operator XJ console command. The PR will auto-restart in the event of failure. Additional operator action is required only if reloading of software is necessary.

Operational Sequences

Operator Action Depress and release front panel initiate (INIT) pushbutton.

Display	Console bell, then LED display of sequential execution of initialization sections 1-6 (01-06 in code LEDs) by each CPU (N000 in data LEDs, where N is the CPU number). Blinking display if errors. See Section 5.2.
Display	LED display of initialization section 7 by one CPU plus console message for loading of operating system: TURN TAPE ON-TURN TAPE OFF-END Blinking LEDs and console messages if error. See Section 5.2 and Section 5.4.17 for tape cassette load procedure.
Display	LED display of initialization section 8 by each CPU (08 N000 in code/data LEDs). Blinking display if errors. See Section 5.2.
Display	Final LED display for successful initialization - 0F 0000. Console message for Debug job initiation: JOB DEBUG Debug job is executing in PR halted (stand-alone) mode. Operator is required to load jobs and initiate job execution. See Section 5.4 for console commands.

5.1.3 Maintenance/Test Operations

For diagnostic testing and maintenance of the IPR, auto-restart and down-line load are normally disabled via the hardware straps. A local I/O channel console with software load capability is implemented. Alternatively a simpler local console (keyboard/printer) may be used if down-line load (strap) is enabled to remotely load the operating system and off-line diagnostic software.

For special maintenance situations, the initialization mode straps may be set (See Section 5.2.3.1), the load-and-go capability (see Section 5.2.3.2) may be used, or the operator GO (see Section 5.4.15) command may be used.

The operational sequence is the same as shown in Section 5.1.2. The operating system is down-line loaded or loaded locally by the operator depending upon the down-line load strap. The final LED display is 0F 0000 or blinking 0F nnnn to define the number (nnnn) of initialization errors. Also if errors, the console message below is printed:

INIT/RSET FAULT

5.1.4 Operator Action Following a Fault

If the PR is configured for auto-restart, initialization is automatically invoked as part of the fault service. If not, the operator must manually restart the system following a fault. This may be done either by executing the restart (RS) command or by depressing the RSET pushbutton. The execute jobs (XJ) command should not be executed following any fault except "TRACE FAULT". Other operator control/monitor console commands may be entered before restarting the system.

5.1.4.1 LED Fault Display

During steady-state operation, hardware faults are displayed in the front panel LEDs in the following format:

Code digit 1:	CPU number
Code digit 2:	Type
Data digits 1-4:	PC, for type > 0
	ST, for type = 0
Type bit 0 on:	Watch dog timer
Type bit 1 on:	Bus response time-out
Type bit 2 on:	Invalid op-code
Type bit 3 on:	Memory parity error
Type = 0:	Unused interrupt

For the hardware faults which cause a failure interrupt, the program counter (PC) when the fault interrupt occurred is displayed. A failure interrupt is not presented to the CPU until the instruction after the one which caused the failure, and by the time it is recognized by the CPU, the program counter (PC) will have been incremented once again, i.e., the PC displayed points to the instruction which is two instructions beyond the instruction which actually caused the hardware failure.

For unused interrupts, the status (ST) after the trap has taken is displayed. The CPU sets the interrupt mask - the least four significant bits of the ST - to a value one less than the interrupt level. For example, a ST = 0001 would indicate that a level 2 interrupt had occurred. Levels 2, 5, 10, 13, 14, and 15 are unused.

Because of the often repetitive nature of hardware faults, the fault service includes a safeguard that only one fault per CPU can occur without restarting the system. If a second one does occur, the CPU takes one of three actions, depending on the strap settings: if auto-restart and down-line load are enabled, power-up initialization is invoked, if auto-restart is enabled but down-line load is not enabled, restart initialization is invoked; if neither auto-restart nor down-line load is enabled, the CPU idles.

5.1.4.2 Console display

All faults, hardware and software, are printed on the console in the format of the message display for the display job (DJ) command shown in Section 5.4.8.

5.2 Initialization

5.2.1 Causes

When power is first applied to the PR, power-up initialization occurs. If power is lost and then re-applied before initialization has completed, the section currently being executed is begun again. After initialization has been completed, a loss and re-application of power causes restart initialization to begin again.

Pushing the INIT pushbutton at any time causes power-up initialization. Pushing the RSET pushbutton causes the initialization section being executed to be begun again. After initialization has been completed, pushing the RSET pushbutton causes restart initialization to begin again.

After initialization has completed, the operator may enter console commands. The IN command is functionally equivalent to the INIT pushbutton; executing it causes power-up initialization. The RS command is functionally equivalent to the RSET pushbutton; executing it causes restart initialization.

In order to re-initialize the system after initialization has completed, push RSET or execute the RS command if you don't need any of the sections of power-up (RAM tests, mapping register tests, or bootstrap load) to be done. Otherwise push INIT or execute the IN command. Do not select INIT or IN if the contents of any of the RAM, e.g., loaded jobs, need to be preserved.

5.2.2 Displays

5.2.2.1 LED display

The format of the front panel LED display during initialization is as follows:

Code Digit 1:	0
Code Digit 2:	Section ID
Data Digit 1:	CPU number
Data Digits 2-4:	Error code (if blinking)

As each CPU starts execution of each section, the section ID and CPU number are displayed. The section IDs are listed in Figure 22.

When an error is detected during initialization, its code is superimposed on the LED display as the last 3 digits of the data LEDs, and the display is blinked for at least 2 seconds. The error codes are listed in Figure 23.

An "F" is displayed in the second digit of the code LEDs and the total number of initialization errors is displayed in all 4 digits of the data LEDs in the last section. If the number of errors is non-zero, the display is blinked.

5.2.2.2 Console messages

The beginning of power-up initialization is announced by the console bell.

Section 7 includes the bootstrap load of the operating system. The console messages and operator actions are described in Section 5.4.16 or Section 5.4.17. Section 9 may include a job down-line load with the accompanying console message:

DOWN-LINE LOAD-END

After initialization has completed, if there were any errors, the following console message is printed:

INIT/RSET FAULT

In any case, the message announcing the beginning of the execution of Debug is then printed:

JOB DEBUG

5.2.2.3 Error Information Stored in Memory

Information about initialization errors is stored in memory. The total number of errors is stored at the location labeled "WSINER". The first error's section ID, CPU number, and error code (i.e., 2 words which are the contents of the front panel LEDs), the CPU's status, program counter, and workspace pointer, and the contents of the 16 workspace registers are stored at the location labeled "ERBFR". The number of times the PR has restarted, without going back to power-up initialization, is stored at the location labeled "WSINRC".

	SECTION ID	SECTION NAME
Power-Up	1	PROM checksum test
	2	RAM test with incrementing test values
	3	RAM test with decrementing test values
	4	Mapping register read back test
	5	Mapping register address memory test
	6	RAM test/initialization with zero
	7[1]	Bootstrap load and checksum generation
Restart	8	OS initialization and OS checksum tests
	9	OS initialization and jobs' load and checksum tests
	A[2]	Error summary

[1] Section ID 7 is displayed only by the first CPU to execute it.

[2] When the number of errors is displayed, the section ID character changes to an F.

FIGURE 22 DEFINITION OF INITIALIZATION SECTION IDS

ERROR CODE	SECTION ID	ERROR DESCRIPTION
100-108	7,9	Load error - see Section 5.4.16 (LED error 10N corresponds to printed error N)
901	4	Bias register 1 did not read back correctly
902	4	Limit register 1 did not read back correctly
903	4	Bias register 2 did not read back correctly
904	4	Limit register 2 did not read back correctly
905	4	Bias register 3 did not read back correctly
906	4	Limit register 3 did not read back correctly
A0X[1]	1,8	Checksum for zone 4 address space X incorrect
AYY[2]	9	Checksum for job YY incorrect
B0X[1]	1,8	Parity or bus time-out error in zone 4 address space X
BYY[2]	9	Parity or bus time-out error in job YY
C00	8,9	Parity error in zone 4 RAM (CPU addresses A810-BFFE, bus addresses FD408-FDFFF)
ZZZ[3]	2,3,6	RAM test error between bus addresses ZZZ00 and ZZZFF
ZZZ[3]	5	Mapping register address memory test error at bus address ZZZ00

[1] The zone 4 address spaces are:

X	CPU Addresses	Bus Addresses
1	F000-FFFE	FF800-FFFFF
2	E000-EFFE	FF000-FF7FF
3	AD42-AE5C	FD6A1-FD72E
4	B320-BE12	FD990-FDF09

[2] Jobs are numbered $05_{16} <= YY <= 28_{16}$

[3] The possible values of ZZZ depend on where RAM is actually installed. Currently, 000 <= ZZZ <= 02B, 040 <= ZZZ <= 04F, or FD4 <= ZZZ <= FDF.

FIGURE 23 DEFINITION OF INITIALIZATION ERROR CODES

5.2.3 Special Test Modes and Procedures

5.2.3.1 Test Mode Straps

Three straps on the Configuration Board control special initialization test modes to aid in debugging:

Bit 13: Loop on error

Bit 14: Write in loop on error

Bit 15: Mini-initialization

The "loop on error" straps (bits 13 and 14) have effect only when an initialization error has been detected. Once an error has been detected, the CPU will loop indefinitely on the set of instructions which caused/detected the error. Errors are displayed only momentarily, rather than for 2 seconds, and the display returns to normal for the loops in which the error is not detected. Each loop is fast enough to allow oscilloscope and/or logic analyzer investigation.

The "write in loop on error" mode (bit 14) is identical to the "loop on error" (bit 13) mode except for the addition of the appropriate write instruction in the loop. For example, the RAM test would add the write instruction:

```
MOV   R0,*R2
```

to the loop containing the read and test instructions:

```
MOV   *R2,R1
C     R1,R0
```

The mini-initialization mode (bit 15) is used when the only desire is to complete initialization in order to use Debug commands like AM, DM, and LD. The mini-initialization mode skips sections 1 through 5. It also displays errors only momentarily (although the total number of errors, if any, is still displayed at the very end).

The following rules apply to the interaction of the enabled straps:

- Bit 13 (loop on error) overrides Bit 0 (auto-restart)
- Bit 14 (write in loop on error) overrides Bits 0 and 13
- Bit 15 (mini-initialization) overrides Bits 0, 13, and 14

5.2.3.2 Bootstrap Load and Go

Instead of loading the O.S. cassette tape, you can load other absolute, i.e, not relocatable, programs for immediate execution. Such a load and go program is identified by adding the entry point address as the operand of the END statement. This produces a "1" tag followed by the address in the object code. The loader, having read this "1" tag and the address, branches to the address after finishing the load.

You should be aware of the environment in which a load and go program will execute. The CPU that does the load (CPU A) is the one that will execute the program. The workspace in which execution will begin is "WSTRMS", currently at address $AACC_{16}$. If you desire a different workspace, do a LWPI instruction. Interrupts will be masked off; doing a LIMI instruction will change that.

You should also be aware of the other CPU (CPU B), the one that didn't do the load. It will stay in the ABS loop at "IN3000" until the CPU executing the load and go program (CPU A) does a RSET instruction or changes workspace "WSIN" register R7 to a negative value. After CPU B is released from the ABS loop, it will display section 1 and its CPU number in the LEDs and then branch to the contents of workspace "WSIN" register R10 (B *R10). This means that the load and go program should set WSIN's R10 to the entry address appropriate to CPU B before setting WSIN's R7 negative.

You can write a source load and go program, assemble it, and dump the object file on cassette, or you can just create the machine code on cassette directly. The cassette should be formatted according to the instructions in the following paragraphs.

Every record should start with a line feed and an asterisk (*), except the first record, which should start with a less-than sign (<). Every record should end with an "F" and a carriage return, except the last record, which doesn't need the "F".

The first record should have a " < ", followed by five zeros and the 8 character name of the program, ended by an "F" and a carriage return.

All other records, except the next-to-last and last, after the line feed and "*", should have a "9" and a four-hex-digit absolute address at which to start loading consecutive data, followed by up to 12 groups of "B" and a four-hex-digit datum to load, ended by an "F" and a carriage return. For example, the record

*9AD40B02E0BA880F

would cause the data $02E0_{16}$ and $A880_{16}$ (LWPI > A880) to be loaded at locations $AD40_{16}$ and $AD42_{16}$, respectively.

The next-to-last record, after the line feed and "*", should have "1" and the absolute entry address of the program, ended by a carriage return.

The last record should have a line feed, "*", colon ":", and carriage return.

5.3 Operator Interface for Console I/O

The console I/O routines implement job requested output to the local (RS-232 interface) console and allow an operator to address input to different jobs. The single console is time-shared between the multiple jobs of the IPR.

If several jobs were allowed to interleave their outputs without any delimiting character or phrase, the resulting display could be very confusing to the operator. For this reason, the console I/O routines will insert the phrase "JOB XYZ" before job XYZ's output is printed. If job XYZ makes several output requests before another job requests an output, the phrase "JOB XYZ" is only printed before the first output.

The console I/O routines allow an operator to type an arbitrary character string and indicate which job is to receive the input. To designate an input for a particular job, the operator must type "@" followed by the job name, a space, and the desired input string terminated by a control-E (ETX) or a carriage return (CR). For example, in order to give the input string "PQR" to job "XYZ", the operator would type

@XYZ PQR (ETX or CR)

The job name may be abbreviated to any number of characters. If the abbreviated name is not unique, the console I/O routines will give the input to the first job requesting input that matches the abbreviated name. For example, suppose the list of jobs requesting input contained the following jobs in this order: XYZ, ABC, ACB. The job XYZ could then be referenced in any of these ways:

@X PQR (ETX or CR)

@XY PQR (ETX or CR)

@XYZ PQR (ETX or CR)

The job ACB could be referenced in two ways:

@AC PQR (ETX or CR)

@ACB PQR (ETX or CR)

If the name ACB were abbreviated to simply A, the input would be given to job ABC.

The specific job to receive input need not be explicitly stated for every input. If the first character the operator types is not "@", then everything typed is given to the last job that was explicitly specified. On initialization or when the PR is halted, the job Debug is assumed to have received the last input.

While inputting characters, the operator has limited editing capability. The rub-out (DEL) character deletes the last character input by the operator and is echoed by "\ ". Several characters can be deleted by repeatedly hitting DEL. To delete all characters previously entered and exit the input mode, the operator should type ESC, which is echoed by "$", carriage , and line feed.

If the operator attempts to input a character and is echoed by "$", carriage return, and line feed, then all three input buffers are busy, i.e., the buffers have not yet been released by jobs that previously received input.

If an operator attempts to input a line of text and "?" is printed, then either the job name entered is not in the system job list or the job specified will not accept input.

5.4 Operator Control/Monitor (DEBUG) Commands

The following paragraphs describe the operator console commands used to control and monitor PR operation, to support PR software development, and to support PR hardware testing and maintenance.

5.4.1 Operation

The Debug job may be executed in normal PR operation or PR halted (stand-alone) operation. In normal operation, Debug is executed like any other job. It may be called by the operator or by another job. It shares the I/O channel console with other jobs.

When the PR is halted, Debug is executed in a single CPU independent of the normal scheduler. Console and Radio/1822 DMA I/O are executed in a polling non-interrupt driven mode. The console is dedicated to the Debug job. No other jobs are executed.

The PR halted mode is invoked by hardware or software failure, upon completion of PR initialization, upon execution of a software trace breakpoint, or by operator or another jobs HJ, TJ, IN, or RS console command. Normal execution is invoked by the operator XJ command or by automatic XJ if auto-restart is enabled and jobs are resident. If the PR is halted by the operator HJ command or by an operator specified breakpoint then the operator must input the XJ command to resume normal operation.

All operator commands are valid when the PR is halted. Some commands are invalid in normal operation.

5.4.2 Operator Command Input

The operator in normal PR operation must direct Debug command input to the Debug job by typing

@DEBUG

prior to the command. This is done once and need not be repeated until the operator wishes to input to another job. A truncated job name may be used as long as it is unique.

The input "@DEBUG" is not required when the PR is halted (stand-alone) and will if input be interpreted as a command error. All console input is given to Debug when the PR is halted.

An operator begins with a two character command code, sometimes followed by one or more operands, and terminated by the single character Control-C (ETX) or carriage return. Debug provides for command chaining. The ";" separates individual commands. ETX or carriage return terminates the single or set of chained commands. One or more ASCII space characters are used to separate command codes and operands in command strings. Input may be edited, as described in Section 5.3, until input of ETX or carriage return. Operator input for a single command or set of chained commands is limited to a single console line.

Debug interprets and executes the command. Upon completion of command processing a carriage return, line feed, line feed is output. If a command input syntax error is detected, the message "CMD?" is printed.

All numeric operator input and numeric display is in hexadecimal 0-9, A-F.

5.4.3 Restart Initialization (RS) Command

This command is equivalent in operation to depression of the front panel restart (RSET) pushbutton. A RSET instruction is executed to invoke restart initialization. Command chaining after RS is permitted. RS command is always valid. Operator input is shown below:

RS

5.4.4 Power-Up Initialization (IN) Command

This command is equivalent to depression of the front panel initiate (INIT) pushbutton. Commands chained beyond IN are lost. IN command is always valid. Operator input is shown below:

IN

5.4.5 Halt Jobs (HJ) Command

This command is used to halt execution of all PR jobs. Any job currently executing (busy) is checkpointed. Normal operation is suspended and the PR enters the PR halted (stand-alone) mode of operation. This command is always valid. Operator input is shown below:

HJ

5.4.6 Terminate Jobs T(J) Command

This command discards all jobs by clearing the system job index and system job list of all entries. This command is only valid when the IPR is halted. Operator input is shown below:

TJ

5.4.7 Execute Jobs (XJ) Command

This command initiates normal PR operation. All idle or halted jobs are called. This command is valid only when the PR is halted. Operator input is shown below:

XJ

Automatic execution of this command is indicated by the console message shown below:

AUTO-XJ

5.4.8 Display Jobs (DJ) Command

This command is used to display the state of an operator selected job (or CPU), or all jobs. It is used automatically to describe a job or CPU halt due to failure or trace breakpoint. Further, the job specified by the operator or halted job defines the job (or CPU)

referenced in other commands. The commands which relate to a specific job are DM, AM, DR, AR, and ST.

The three operator command input options are shown below. These commands are always valid.

DJ

DJ job name

DJ ALL

If the job name is omitted the currently selected job is displayed. The one to five character name specifies the desired job. "ALL" requests the display of all resident jobs in sequential order. Additionally the ALL option displays the PR ID and current operational mode as shown below:

PR I.D. XXXX HALTED

PR I.D. XXXX

The job or halted CPU state is displayed as shown below:

JOB job name

state reason CPU n ST: XXXX PC: XXXX WP: XXXX

The "job name" defines the job or CPU. The CPU name (CPU 1, CPU 2, etc.) is printed instead of a job name for PR halts which may not be associated with a system job.

The "state" describes the job's current execution state, as follows:

BUSY	Job busy executing
HALTED	Job halted due to hardware failure, job detected failure or software trace breakpoint
SPNDED	Job execution suspended
CKPTED	Job execution checkpointed
IDLE	Job has nev r executed or has been re-initialized

The "reason" describes why a job is halted, as follows:

TIMER FAULT	Watch dog timer elapsed
BUS TIMEOUT FAULT	Bus response timeout
INV OPC FAULT	Invalid instruction operation code
PARITY FAULT	RAM or PROM memory parity error
TRACE FAULT	Software trace breakpoint trap
FAULT	Job detected fault invoked by "XOP @XHALT,JBCTRL" instruction

INTR FAULT Invalid (unused) interrupt fault

CPU n, where n equals 1, 2, 3, etc., identifies the CPU which is executing or has last executed the job.

XXXX respectively describes the current contents of the non-busy job's CPU status (ST), program counter (PC), and workspace pointer (WP) registers. The PC defines the address of the next instruction to be executed.

XXXX is an absolute 16-bit CPU address or status register value. A job's zone 2 address for the PC and WP register content is displayed as XXXXR. R following the address indicates that the relocation constant has been subtracted from the absolute address before display. XXXXR defines the job address as it appears in the source/object program listing (see use of XXXXR address specification in AM/DM commands).

5.4.9 Display Memory or Peripheral I/O (DM) Command

This command is used to display the contents of selected RAM, PROM, or peripheral device registers. The operator may display single locations or a series of locations between specific address limits. The operator may specify the addresses of locations to be displayed as a 20-bit absolute bus address, as a 16-bit absolute bus address, or as a 16-bit CPU byte address relocated to a specific job's zone 2 address space. Example of the operator input options and resulting display are shown below. These commands are always valid. Operator input is underlined.

```
DM F000
 F000 AAAA

DM E124 E128
 E124 0431 C006 072C

DM 400R
 0410 0000

DM 0W 8W
 00000 0100 B67E 0100 B686 0100 B67E 0080 B6EC
```

The first example illustrates the display of a single absolute 16-bit CPU byte address $F000_{16}$. The second example illustrates the display of a series of locations $E124_{16}$ to $E128_{16}$. The third example illustrates use of the absolute 16-bit CPU byte address relocated in zone 2. The character "R" immediately following the address specifies this option. This option allows the operator to input an address from the job source/object listing and have it relocated to its absolute value in zone 2. The last example illustrates use of the absolute 20-bit bus address option.

Leading zeros in all addresses may be omitted. Mixed address formats in a single command are not permitted. The 16-bit CPU addresses are incremented by 2, while 20-bit absolute addresses are incremented by 1 for each consecutive location.

5.4.10 Alter Memory or Peripheral I/O (AM) Command

This command is used to alter (modify) the contents of selected RAM memory or peripheral device interface registers. The operator may modify single locations or a series of locations beginning at a selected address. Locations in a series may be selectively skipped. As in the DM commands, the selected address may be an absolute or relocated 16-bit CPU byte address, or an absolute 20-bit bus address. The operator input options are illustrated below. These commands are always valid.

```
AM F000 123

AM F000 1234 S 4567

AM 300R 460 310R

AM F0001W ABCD
```

Line 1 modifies absolute 16-bit CPU address F000 to contain 0123_{16}. All data input is right justified in one 16-bit word. Line 2 modifies locations $F000_{16}$ and $F004_{16}$ to contain 1234_{16} and 4567_{16}, respectively. The character "S" causes, in this case, location $F004_{16}$ to be skipped.

Line 3 modifies two locations, 300_{16} and 302_{16}, relocated to the selected job's zone 2 address space. The data 0310_{16} will be modified by the value of the job's zone 2 relocation displacement. Line 4 modifies absolute bus location $F0001_{16}$ to contain $ABCD_{16}$.

Data operands or skip location characters are delimited by one or more space characters. Input is limited to a single console line. Leading zeros in address or data operands may be omitted. Consecutive data operands modify consecutive locations from the selected start address. A 16-bit CPU byte address increments by 2, a 20-bit bus address increments by 1 to address the next consecutive location.

5.4.11 Display Job Registers (DR) Command

This command is used to display the contents of a selected job's registers: status (ST), program counter (PC), workspace pointer (WP), or workspace registers (R0-RF) defined by the workspace pointer. A specific job is selected by the Display Job (DJ) command. The operator input options are shown below. These commands are always valid.

DR ST

DR PC

DR WP

DR Rn_1

DR $Rn_1 Rn_2$

The numbers n_1, and n_2 are hexadecimal numbers, i.e., 0-9 or A-F.

Lines 1, 2, and 3 illustrate the operator input to display respectively the selected non-busy job's ST, PC, and WP registers. The display via printed messages is identical to the Display Memory (DM) command display where the first column displays the address and the second column the contents of the selected register.

Lines 4 and 5 illustrate respectively the operator input to display the contents of a single workspace register or a consecutive series of registers Rn_1 through Rn_2. As above, the display via printed messages is identical to the Display Memory (DM) command display.

5.4.12 Alter Job Registers (AR) Command

This command is used to alter (modify) the current the current contents of a selected job's registers: status (ST), program counter (PC), workspace pointer (WP), or workspace registers. A specific job is selected by the Display Job (DJ) command. The operator input options are shown below. These commands are valid only when the PR is halted.

AR ST F

AR PC 300R

AR WP 0200

AR Rn F123

AR Rn 1234 5678

The number n is a hexadecimal digit, i.e., 0-9 or A-F.

Lines 1, 2, and 3 illustrate the operator input to modify respectively the contents of the ST, PC, and WP registers to contain $000F_{16}$, 0300_{16} plus job relocation displacement, and 0200_{16}. A single register may be modified per operator command. Line 4 illustrates the operator input to modify a single workspace register. Line 5 illustrates the operator input to modify a series of consecutive workspace registers, beginning at the specified register. Data input formats are identical to data input for the Alter Memory (AM) command.

5.4.13 Set Trace Breakpoint (ST) Command

This command is used to select one to six software instruction trace breakpoints. The operator input is shown below. These commands are valid only when the PR is halted.

ST E100

ST 300R 310R 314R 500R 504R 506R

ST S 320R 0

Line 1 illustrates the operator input to define the first trace breakpoint at location $E100_{16}$. A previously defined first trace breakpoint is discarded. The remaining five breakpoints are unchanged. Line 2 illustrates selection of all six trace breakpoints. All breakpoints, in this case, are relocated to the selected job's zone 2 address space. the job is selected by the Display Job (DJ) command. Line 3 illustrates the operator input which skips the first breakpoint (breakpoint remains as previously defined), defines the second breakpoint, and clears the third breakpoint. All breakpoints may be cleared by use of the Clear Trace (CT) command.

Guidelines for selection of trace breakpoints are listed below:

1. Trace breakpoints for more than one job may be concurrently defined. Previously selected breakpoints may be cleared or redefined regardless of previous job selection.

2. Trace breakpoint word aligned addresses must define the first word of multiword length instructions. They must define an instruction (not data) address.

3. Multiple trace breakpoints must not define the same address.

4. Trace breakpoints may be selected in reiterative program sequences as long as continuation address is included. Trace continuation is described below.

5. The 20-bit bus address (W) form may not be used to define trace breakpoints.

The SET Trace (ST) command inserts breakpoints. The instruction at the selected address is saved and replaced by an "XOP R0,TBPT" ($2C80_{16}$) instruction. Job execution is initiated with the XJ command. Execution continues normally until a trace breakpoint is encountered.

Upon execution of the breakpoint instruction, the operating system captures the machine state of the trace halted job, halts execution of all other jobs, and calls the Operator Control/Monitor (Debug) job which displays the state of the trace halted job as shown in Section 5.4.8. Other debug job commands may be used to further investigate the state of the PR at the trace breakpoint.

Software execution may be continued from the trace breakpoint location by use of the Execute Jobs (XJ) command. When the job initiate address (current PC) equals a selected breakpoint, a continuation breakpoint is created at the next instruction location from the breakpoint if this location is not also a selected breakpoint. The breakpoint location is restored. The operating system regains control via the continuation breakpoint, restores the continuation location, and inserts the breakpoint instruction at the original breakpoint. Execution is then continued normally from the continuation breakpoint location.

5.4.14 Clear Trace Breakpoint (CT) Command

This command is used to clear all previously selected trace breakpoints. Original instructions at the trace breakpoints are restored. The operator input is shown below.

CT

This command is only valid when the PR is halted.

5.4.15 Execute Operator Routine (GO) Command

This command specifies an entry address and transfers control to an operator specified routine. Operator input is shown below. This command is always valid.

GO 440R

Only absolute or relocated 16-bit byte address entry addresses are permitted. The operator routine, typically defined by an AM command, is entered at the specified address. The routine executes as part of the Debug job.

Only registers R0-R8 may be used if a new workspace is not defined by the routine. To resume normal Debug execution, the routine should be exited with a "B *R13" ($045D_{16}$) instruction.

The GO command is intended for use in special test/maintenance situations.

5.4.16 Console Load (LD)/Load Verify (LV) Commands

These commands are used to load (LD) object software into the PR, or to verify the correctness of previously loaded software (LV). Operator input is shown below.

LD

LV

These commands are only valid when the PR is halted. If a new (version of an existing) job is to be loaded, the "RS" and "TJ" commands must be executed before the "LD" command to reset the OS and to clear the System Job List. These three commands can be chained, as shown below; if the IPR is strapped for auto-restart, the commands must be chained.

RS;TJ;LD

Load object is typically contained on TI Silent 700 terminal type cassettes. Load object in input from the terminal connected to the PR I/O channel. After operator input of the LD or LV command, the operator is instructed to start data input by the message below.

TURN TAPE ON

To input data from the tape cassette:

1. Insert tape cassette in terminal.
2. Set Record/Playback switch to playback.
3. Set Playback switch to LINE.
4. Set Tape Format switch to LINE.
5. Depress Rewind followed by LOAD/FF to rewind tape and set tape to load point.
6. Depress Playback Control CONT START.

If an error is detected during the load the error message illustrated below is printed.

-ERROR n jobname aaaa

The error number "n" describes the failure as shown below.

Error 0 - A load verification of a relocatable job was requested, but the job did not exist in the system job list.

Error 1 - A relocatable job load was requested, but there were no empty entries in the system job list.

Error 2 - The checksum at the end of an object code record does not match the checksum calculated while inputting the record.

Error 3 - While loading a program, no characters were received for 150 character periods during console load or no load data packets were received for 15 load request periods (75 seconds) during down-line load.

Error 4 - An object code tag not supported by the loader was encountered.

Error 5 - A job checksum error was detected in a job, or a bus time-out or memory parity error was detected while generating and testing the checksum of the load.

Error 6 - Load verification error. The data word from tape did not equal the resident word.

Error 7 - A memory parity error failure was detected.

Error 8 - A bus time-out failure was detected.

The job name is printed for errors detected during loading of the job. The hexadecimal number "aaaa" is a 16-bit CPU address. For errors 6, 7, and 8, "aaaa" defines the memory address in error or address accessed when the error occurred.

Upon completion of the load (with or without errors), the message below is printed.

-TURN TAPE OFF

The operator should stop data input by depressing the STOP switch opposite the CONT START position. After data input has ceased, the message below is printed.

-END

The message resulting from an error free load or load verify is shown below.

-TURN TAPE ON-TURN TAPE OFF-END

5.4.17 Down-Line Load (DL)/Load Verify (DV) Commands

These commands are used to down-line load (DL), or down-line verify (DV) object data via the PR radio or 1822 interface. The commands may be used regardless of the value of the down-line load

enable hardware switch. The commands are only valid when the PR is halted. The operator may specify the name of the load data file. Operator input is shown below.

DL jobname

DV jobname

If a new (version of an existing) job is to be loaded, the "RS" and 'TJ" commands must be executed before the "DL" command to reset the OS and to clear the System Job List. These three commands can be chained, as shown below; if the IPR is strapped for auto-restart, the commands must be chained.

RS;TJ;DL Jobname

The one to seven character jobname if specified is inserted in the load request ROP packet. If omitted any load object file will be accepted (file name in load request ROP is ETX only). After the command input, the message below is printed on the I/O channel console to indicate initiation of the load.

DOWN-LINE LOAD

Errors detected during loading are identified via console message shown below as described in Section 5.4.16.

-ERROR n jobname aaaa

Completion or termination due to error of the down-line load or load verify is indicated by the console message below.

-END

The console message for an error free load or load verify is shown below.

DOWN-LINE LOAD-END

THIS REPORT HAS BEEN DELIMITED AND CLEARED FOR PUBLIC RELEASE UNDER DOD DIRECTIVE 5200.20 AND NO RESTRICTIONS ARE IMPOSED UPON ITS USE AND DISCLOSURE.

DISTRIBUTION STATEMENT A

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.